XIX European TEX Conference
April 29–May 3, 2011, Bachotek, Poland

Grid Typesetting with Inserts Omission

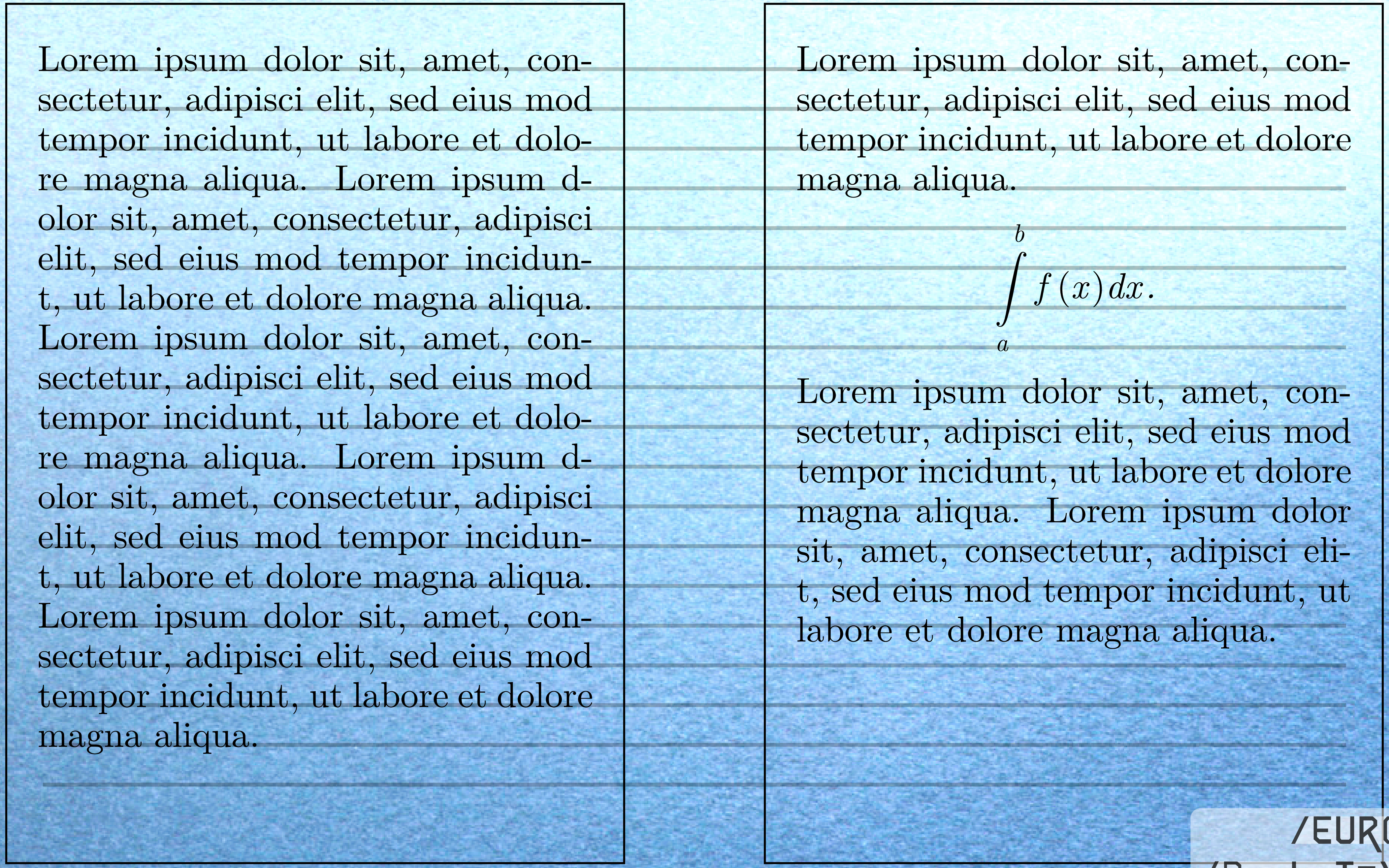
Jacek Czekaj

GUST, Katowice

jacek.czekaj@gmail.com

“Hello!”

/EURO
/BachoTEX
20\\



Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua. Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua. Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua. Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua. Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua.

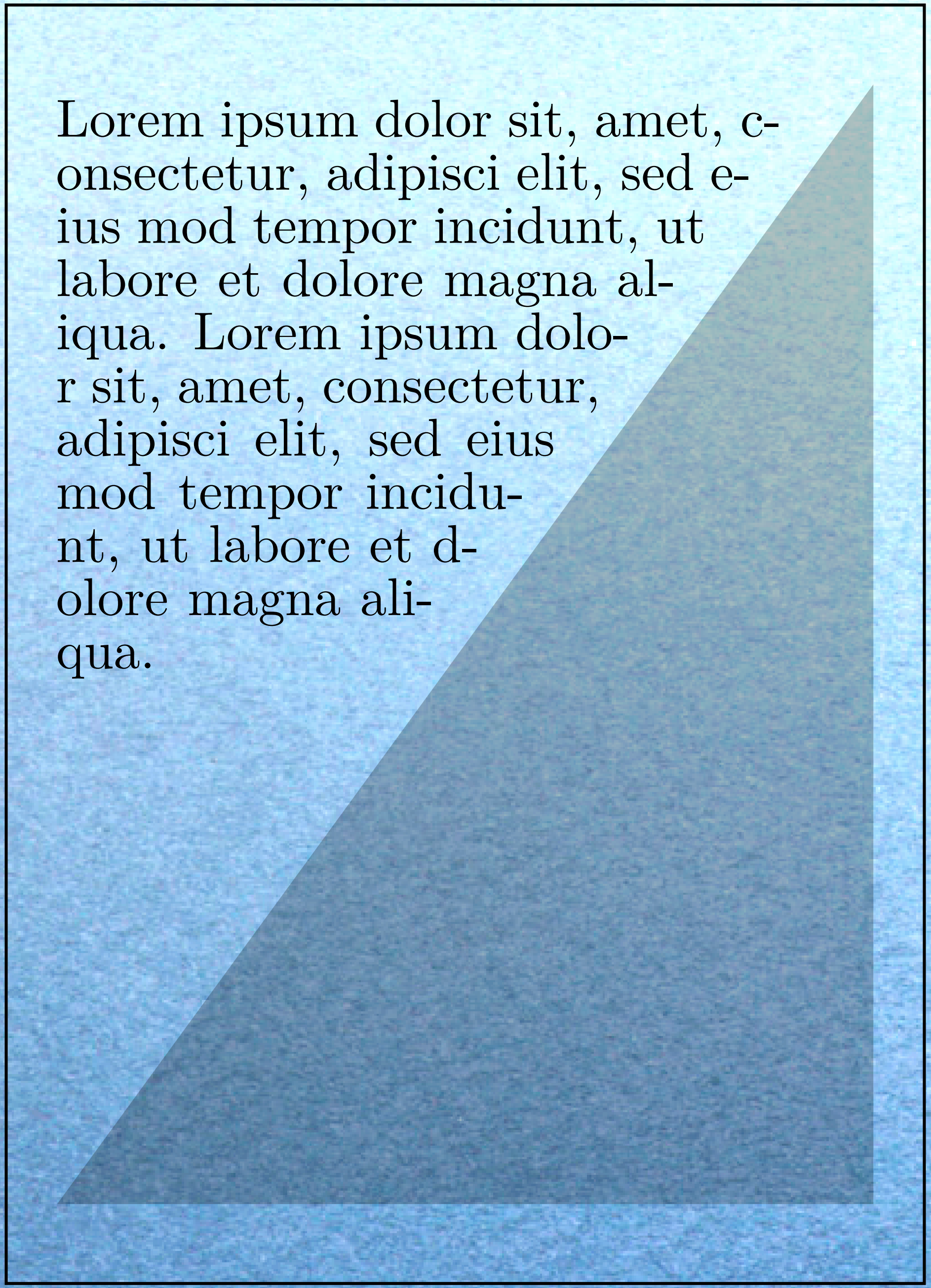
Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua.

$$\int_a^b f(x)\,dx.$$

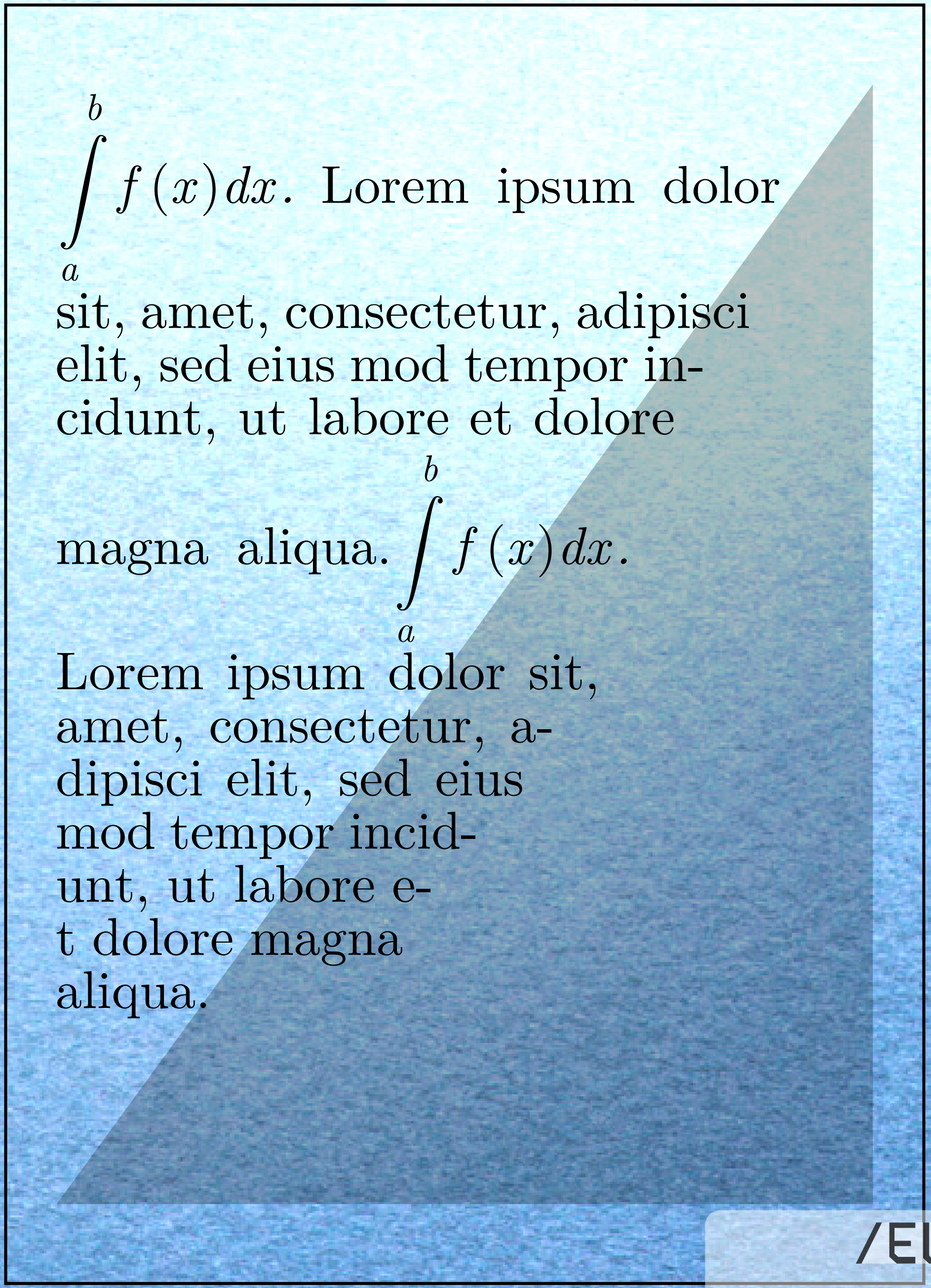
Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua. Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua.

grid typesetting

/EURO
/BachoTEX
20\\



Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua. Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua.



$\int\limits_a^b f(x)\,dx$. Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua. $\int\limits_a^b f(x)\,dx$.

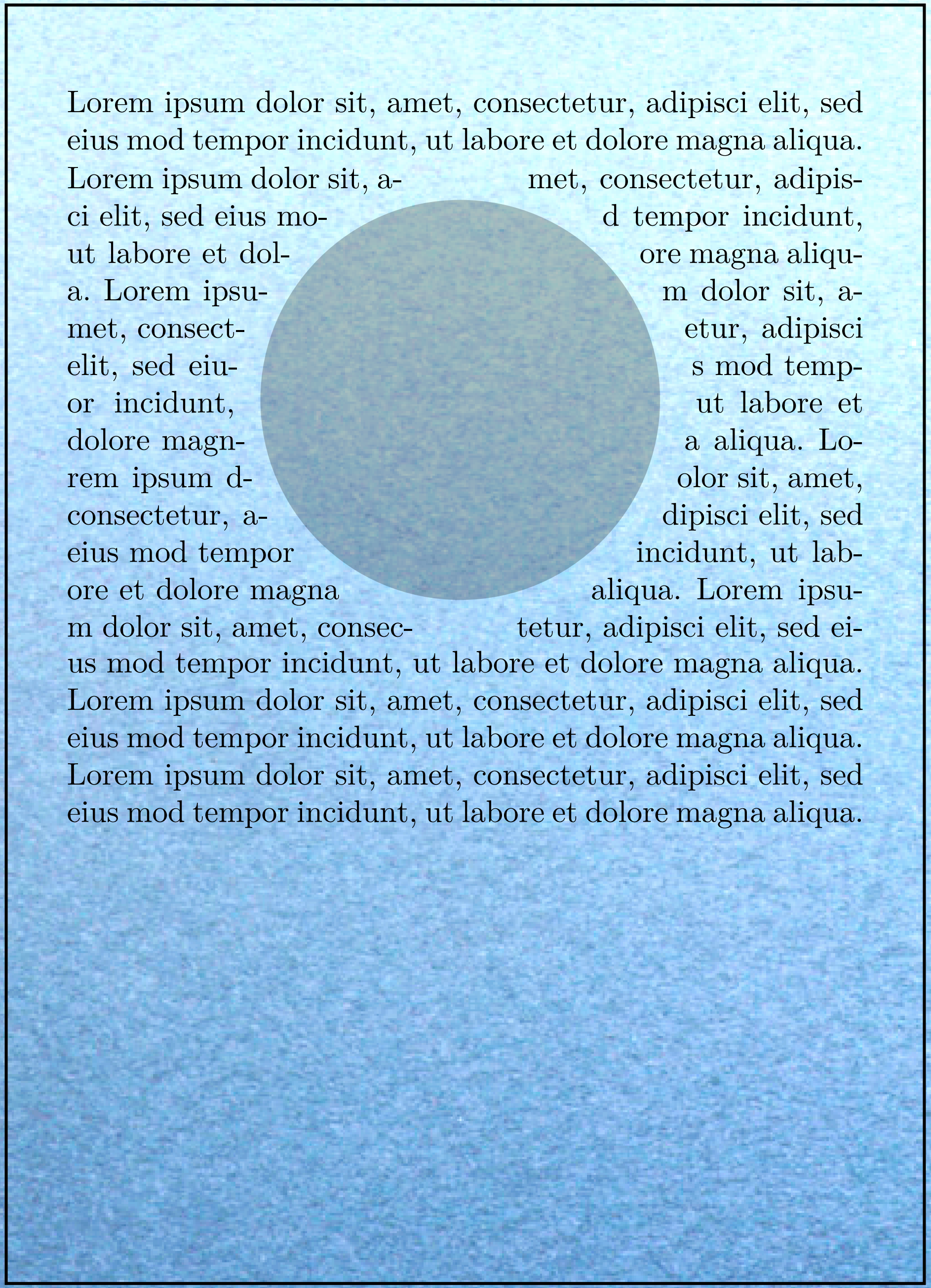
Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua.

inserts omission → \parshape

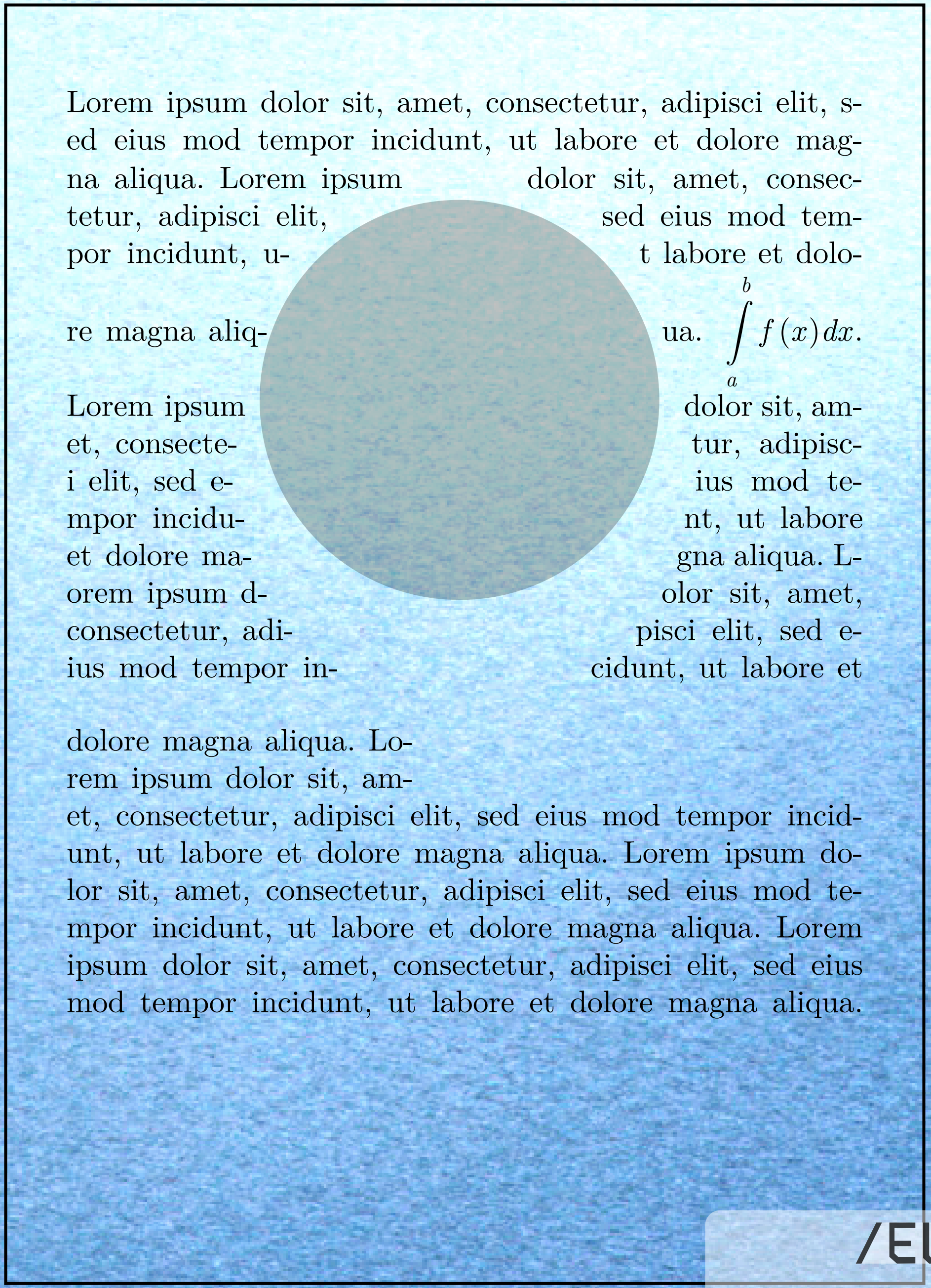
Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua.

$\int\limits_a^b f(x)\,dx$ Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua.

inserts omission → ‘shapepar.sty’



Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua. Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua. Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua. Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua. Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua. Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua. Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua.



Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua. Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua. $\int\limits_a^b f(x)\,dx$.

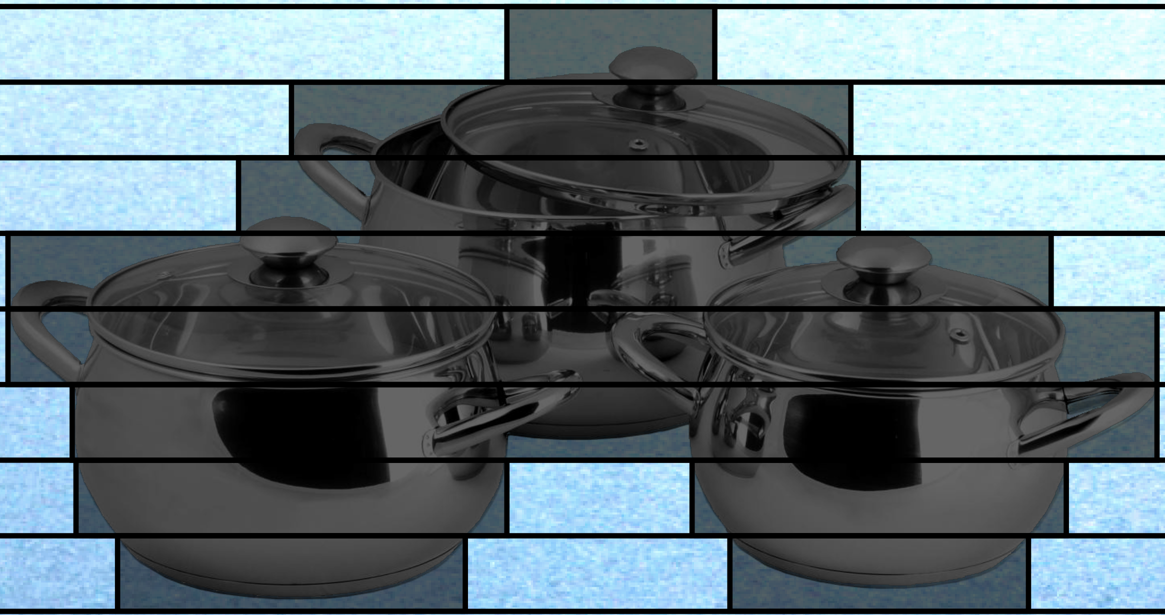
Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua. Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua. Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua. Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua. Lorem ipsum dolor sit, amet, consectetur, adipisci elit, sed eius mod tempor incidunt, ut labore et dolore magna aliqua.

inserts omission → ‘glueball.tex’

text lines, frame

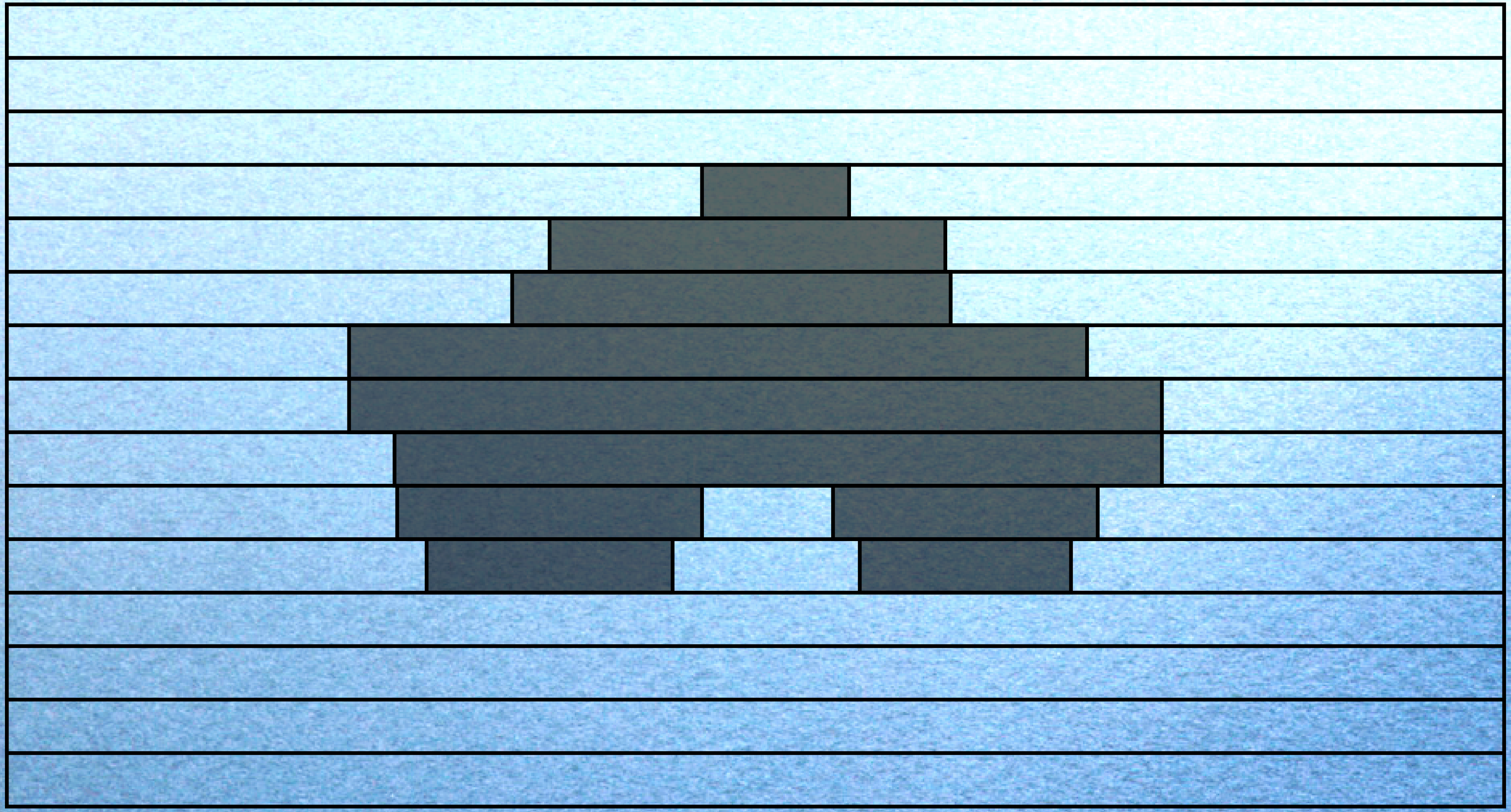
insert

/EURO
/BachoTEX
20\\

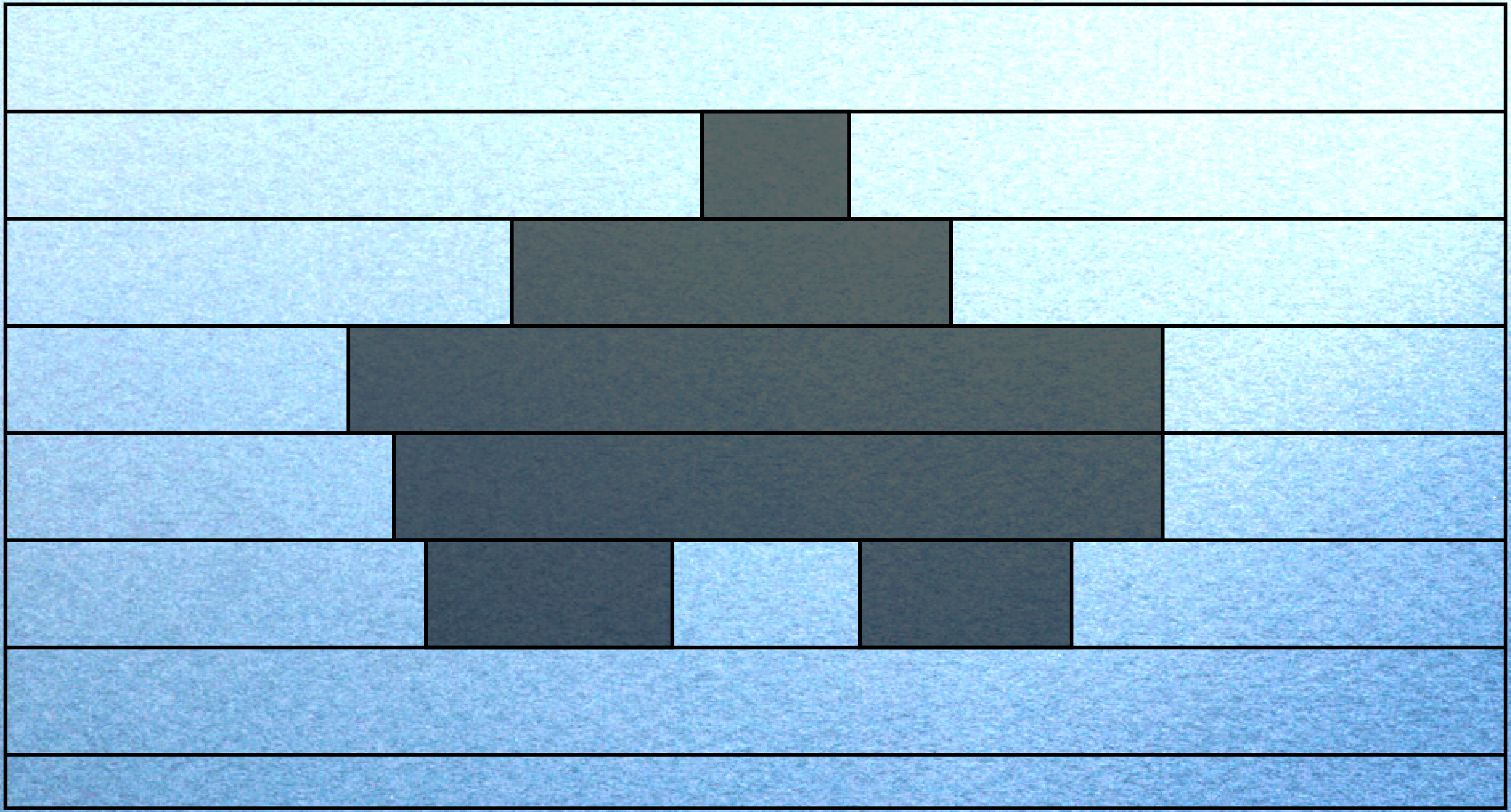


insert omission

/EURO
/BachoTEX
20\\



segments



segments of multi text lines

segment:

☞ begin and end (width)

☞ minimal used space/maximal left space

☞ cost of using

☞ boxes

box:

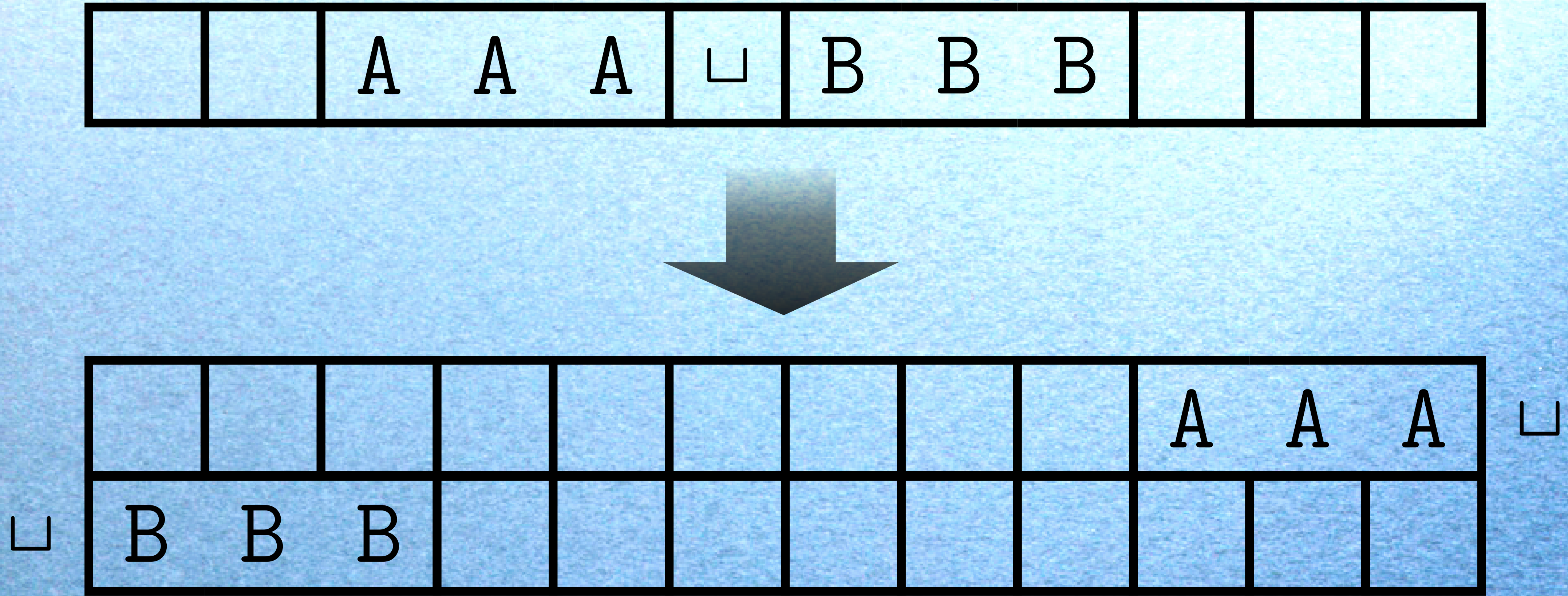
☞ width

☞ height

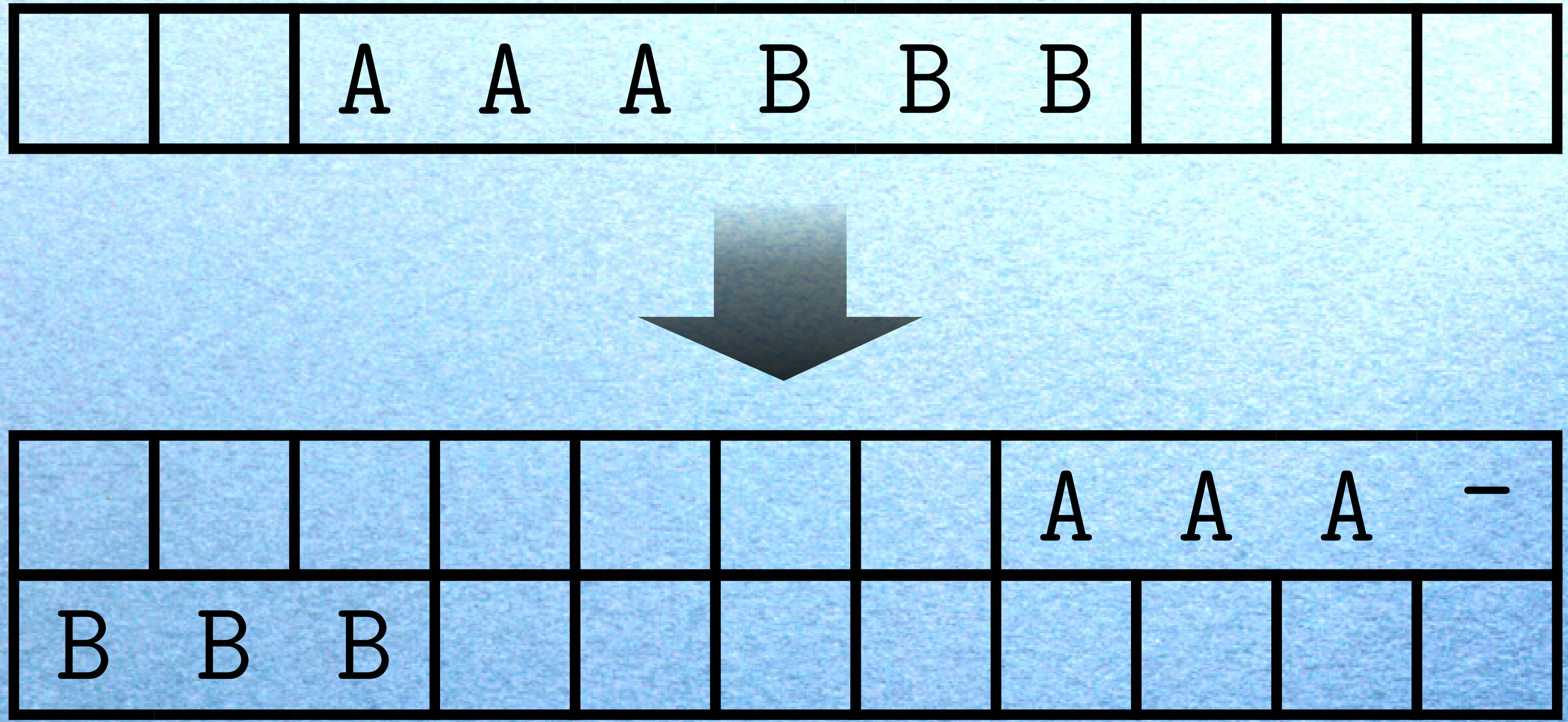
☞ cost of using

☞ “left space absorption factor”

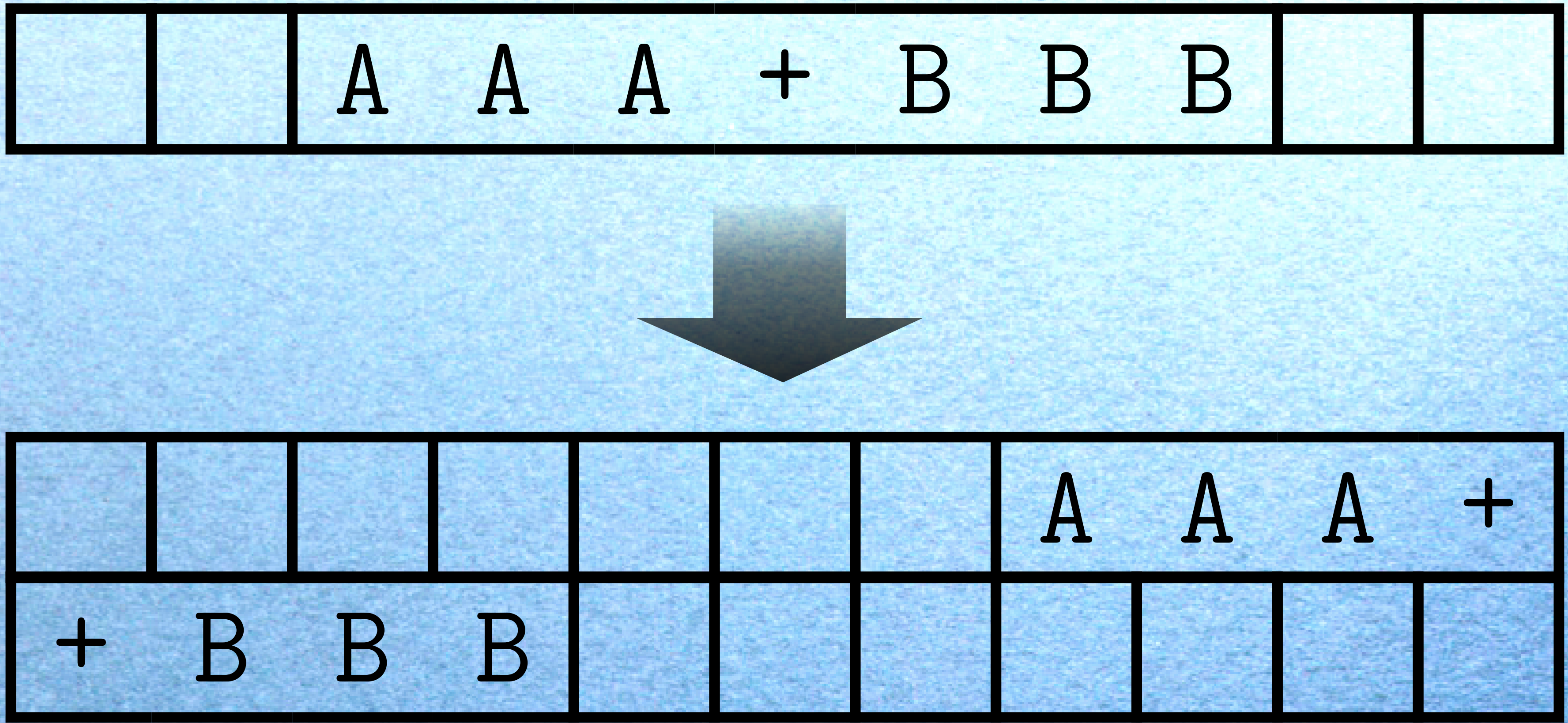
☞ list of pairs of “hyphenation” boxes (each of these boxes has its own “cost of using”)



box → hyphenation boxes → space/non-breaking space



box → hyphenation boxes → discretionary



box → hyphenation boxes → math discretionary one box →

← the box

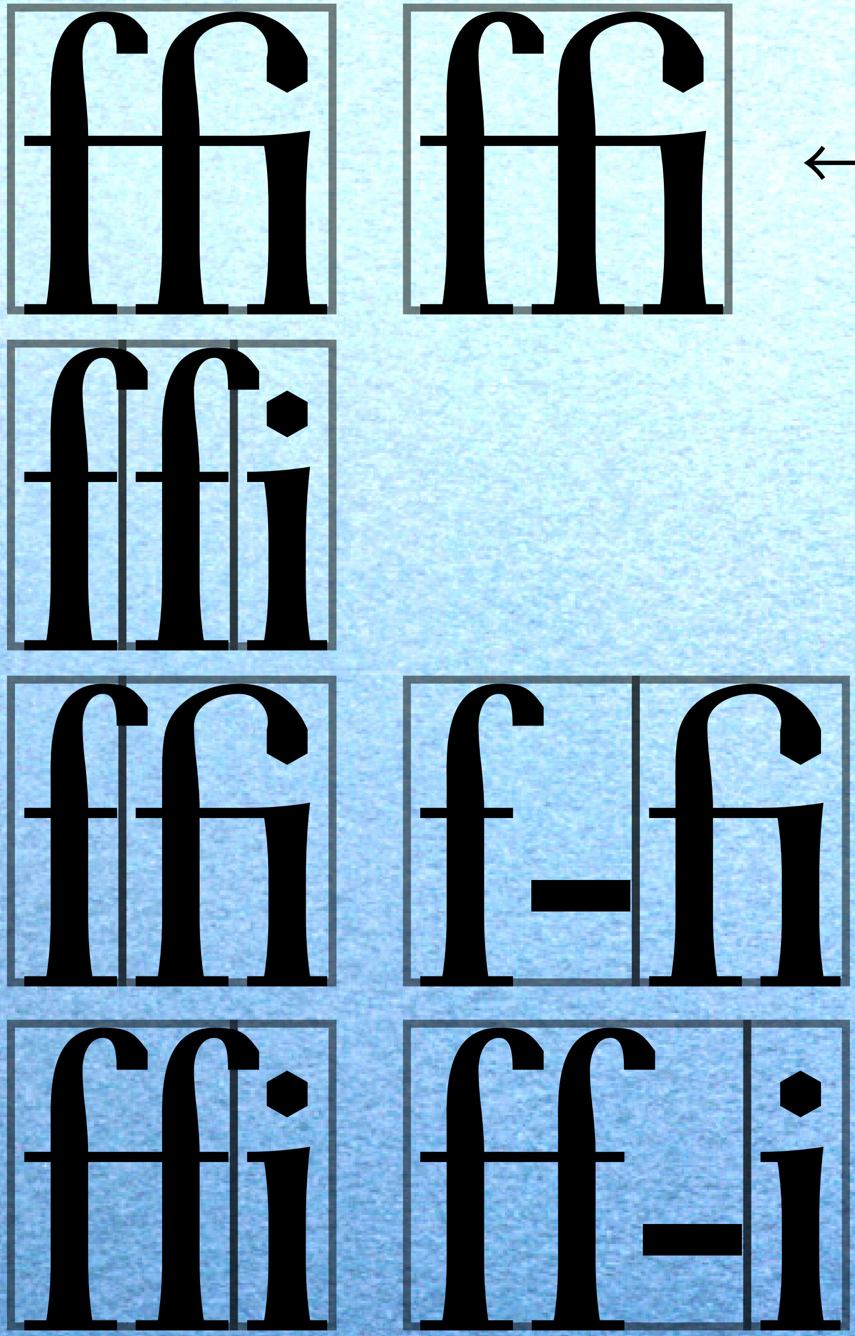
three boxes →

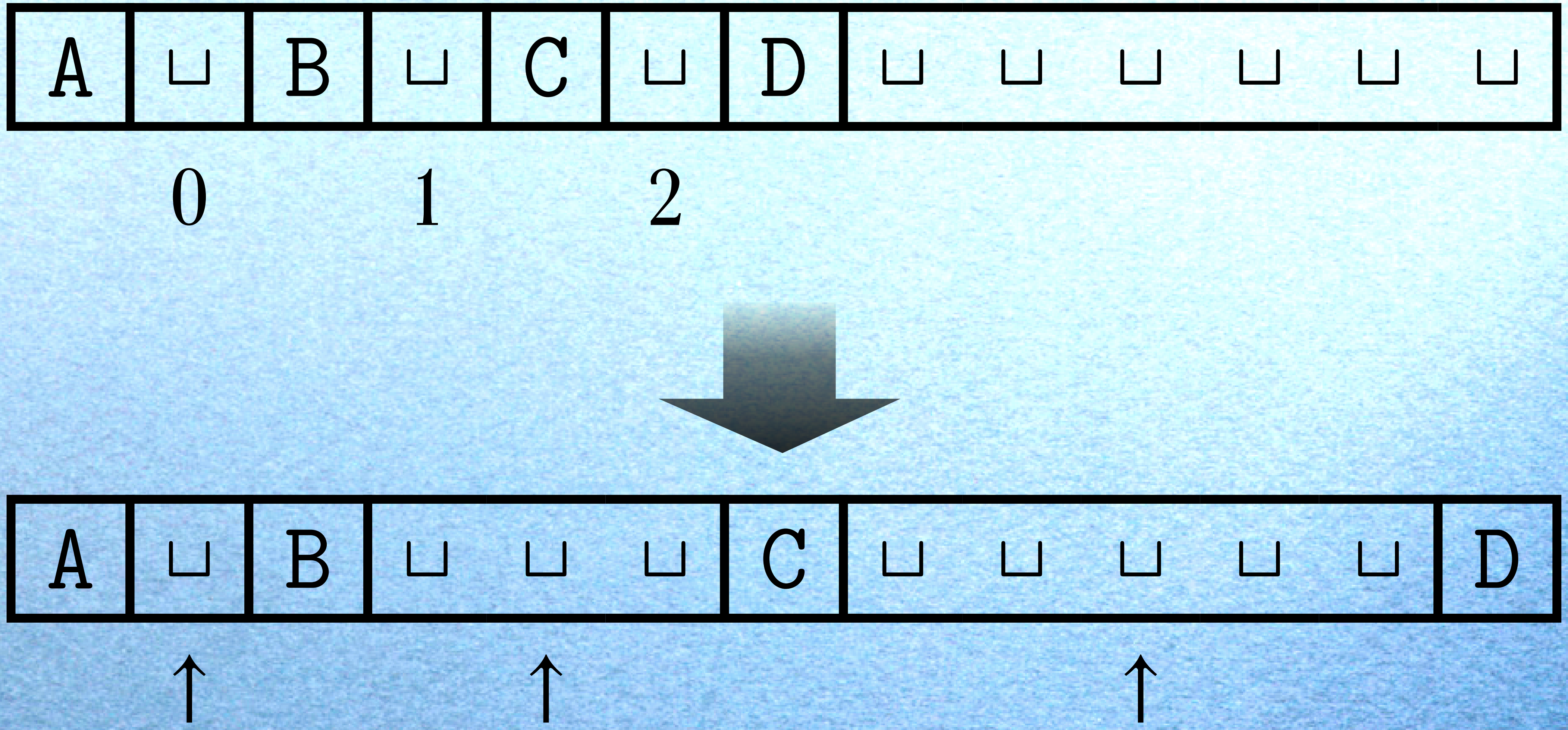
two boxes →

two hyphenation
← boxes related
with the box

two boxes →

another two
← hyphenation boxes
related with the box





$$\frac{0}{0+1+2}\cdot 6=0 \qquad \frac{1}{0+1+2}\cdot 6=2 \qquad \frac{2}{0+1+2}\cdot 6=4$$

box → left space absorption factor

						c
A	A	A	A	A		1
B	B	B	B	B		1
C	C	C	C	C		1
D	D	D	D			0

						c
A	A	A	A	A	A	0
B	B	B				3
C	C	C	C	C	C	0
D	D	D	D			0

algorithm → costs computing

						c	c^2
A	A	A	A	A		1	1
B	B	B	B	B		1	1
C	C	C	C	C		1	1
D	D	D	D			0	0

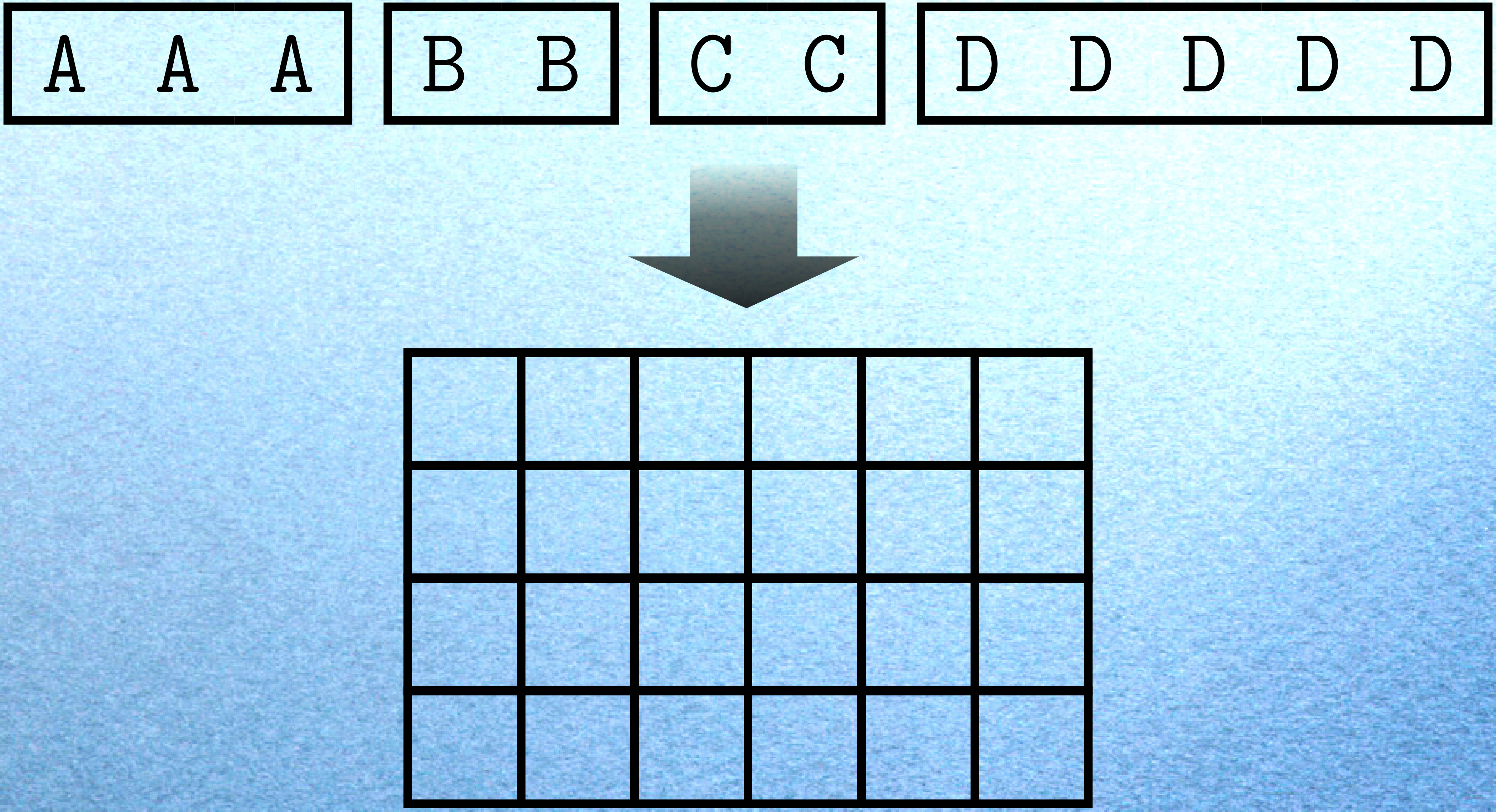
						c	c^2
A	A	A	A	A	A	0	0
B	B	B				3	9
C	C	C	C	C	C	0	0
D	D	D	D			0	0

nodes:

- height, line and segment numbers
- box number
- hyphenation box variant
- current minimal cost
- predecessor node
- current height

special priority queue:

- ☞ consists of multiple priority queues: one for each of the segments of the frame and one additional priority queue
- ☞ push/find a node (for a given h, ℓ, s, b, v): $O(\lg \#N_{\ell,s})$
- ☞ pop a node: $O(\#S)$ (in fact: $O(1)$)



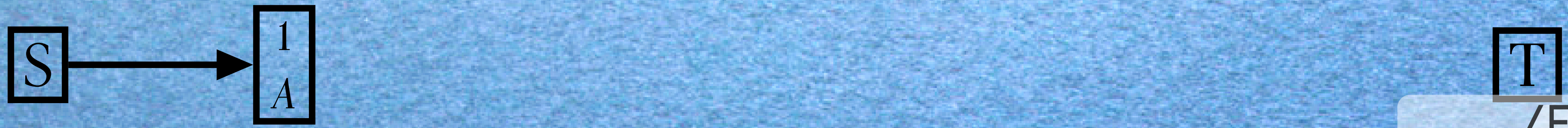
A A A

B B

C C

D D D D D

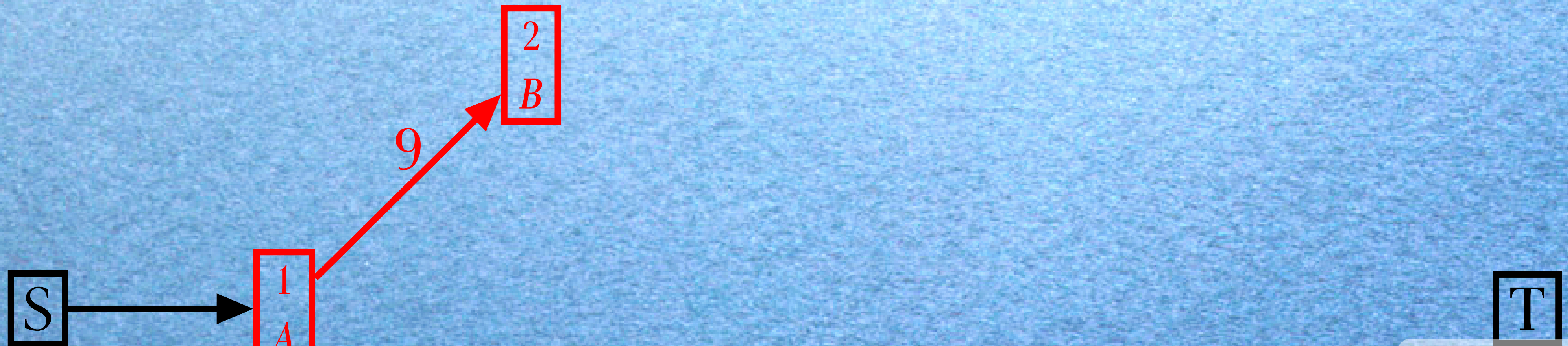
A A A | B B | C C | D D D D D



A A A	B B	C C	D D D D D

A	A	A			

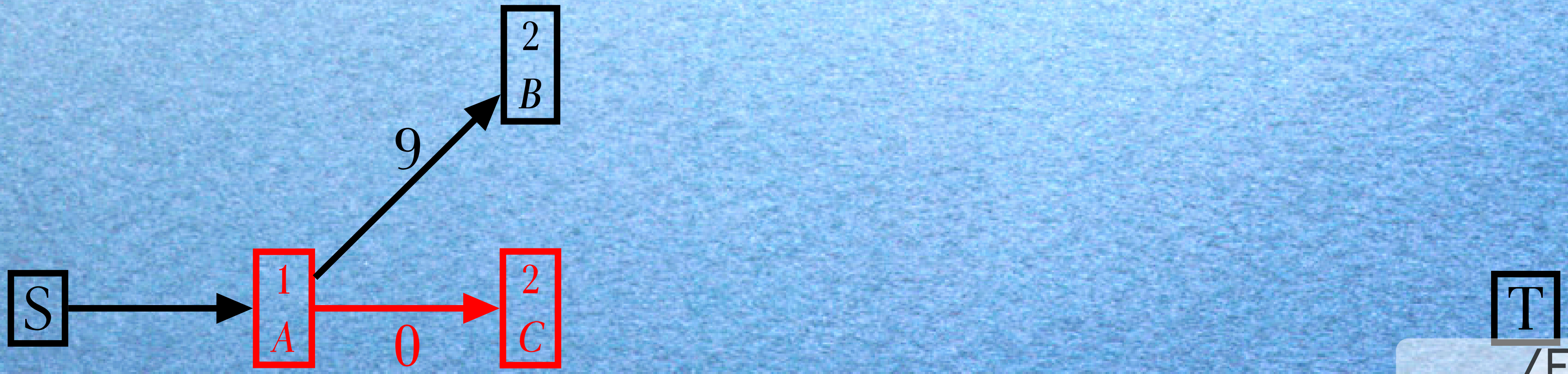
9



A A A | B B | C C | D D D D D

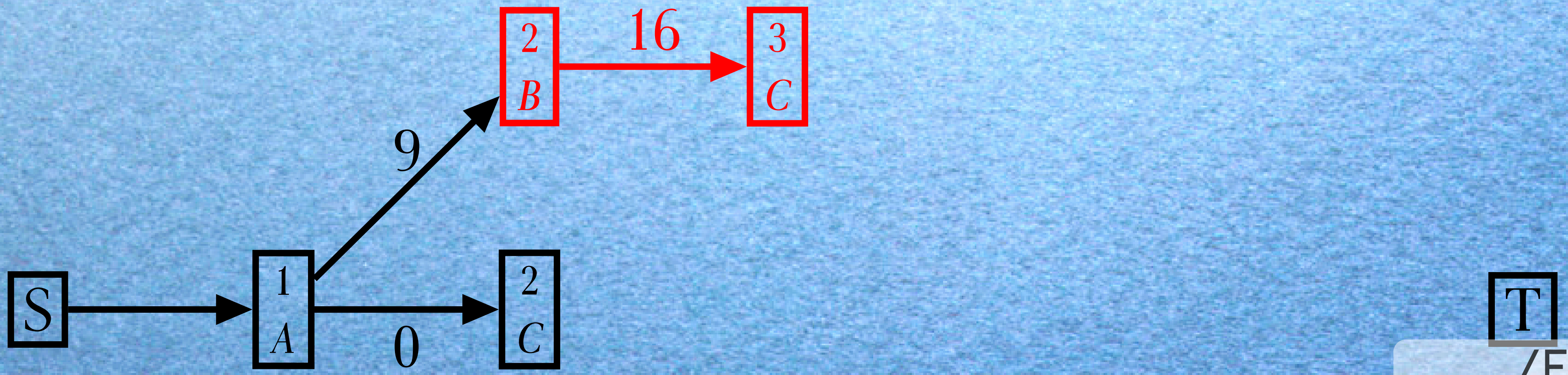
A	A	A	␣	B	B

0



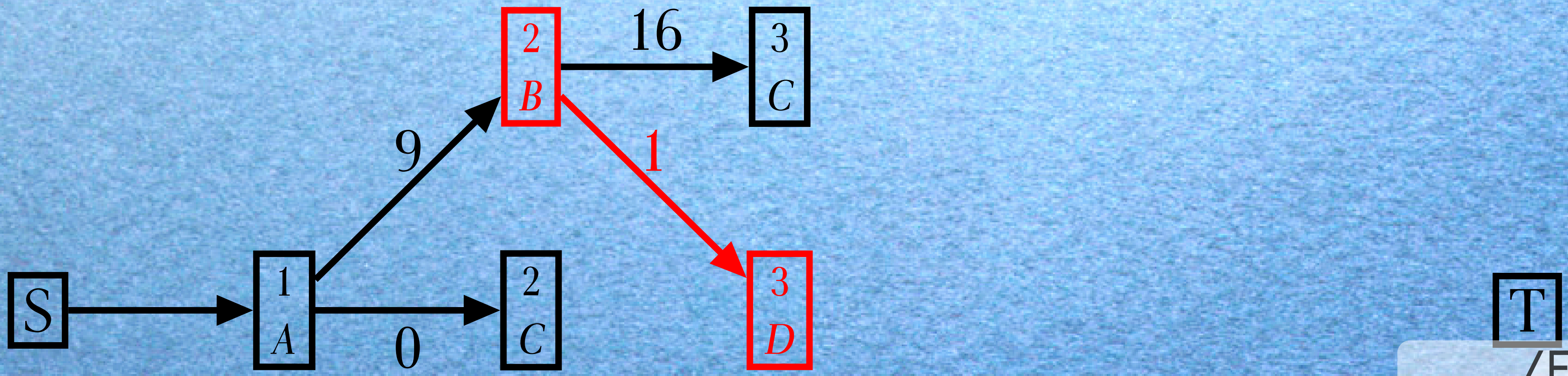
A A A | B B | C C | D D D D D

A	A	A				9
B	B					16



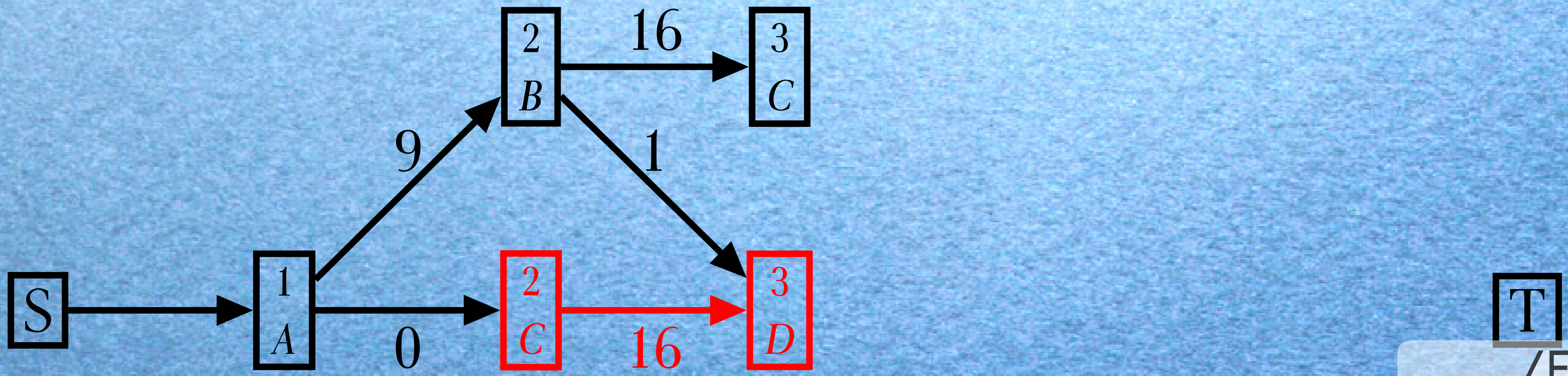
A A A	B B	C C	D D D D D

A	A	A				9
B	B	␣	C	C		1



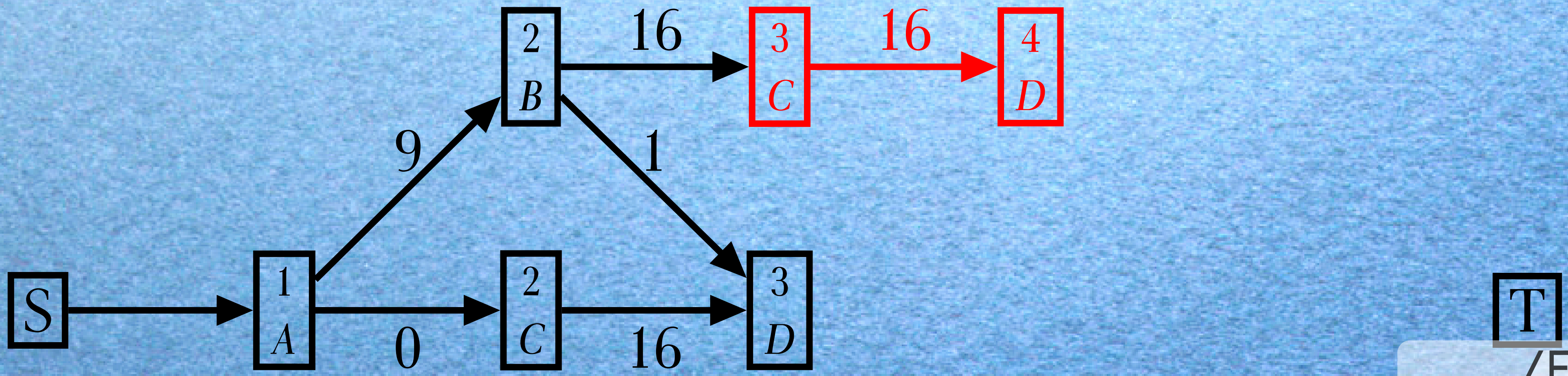
A A A | B B | C C | D D D D D

A	A	A	␣	B	B	0
C	C					16



A A A | B B | C C | D D D D D

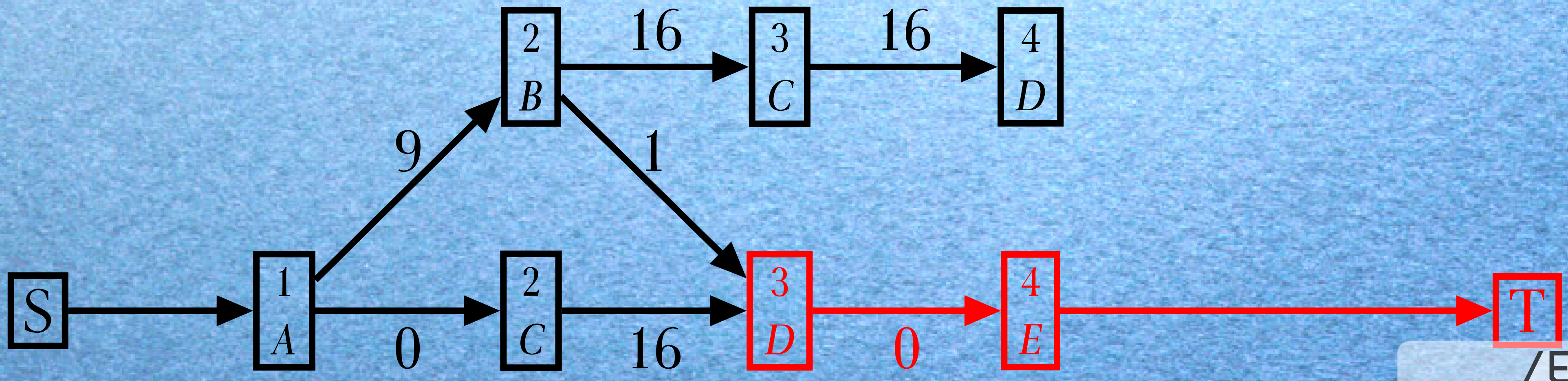
A	A	A				9
B	B					16
C	C					16



algorithm → example 1 → solution

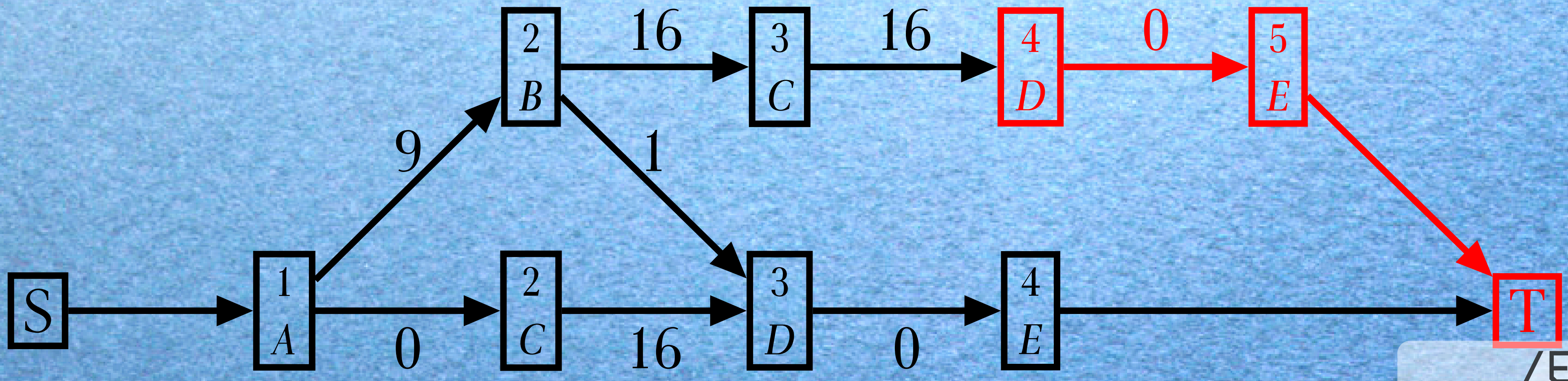
A A A | B B | C C | D D D D D

A	A	A				9
B	B	␣	C	C		1
D	D	D	D	D		0



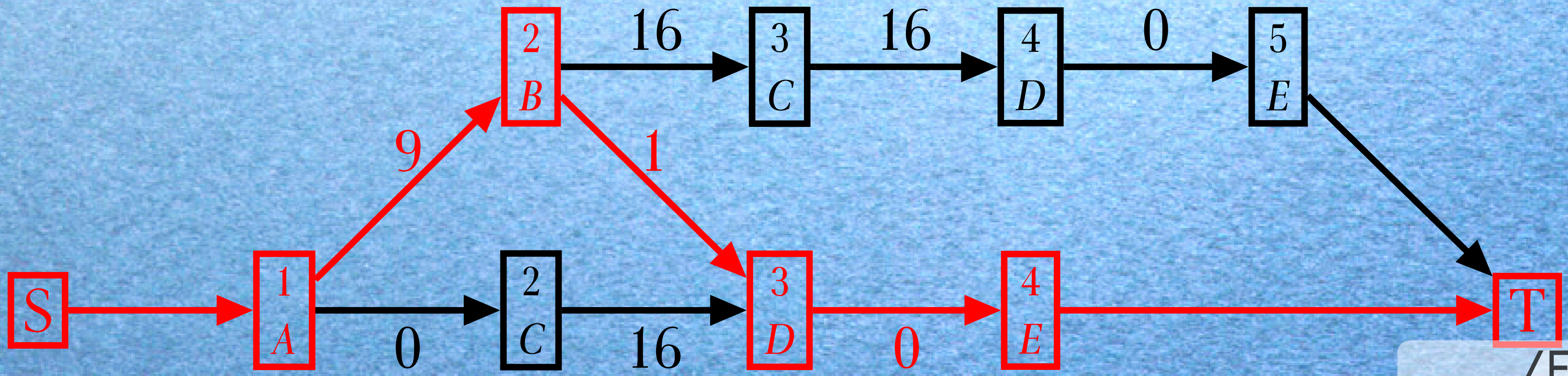
A A A | B B | C C | D D D D D

A	A	A				9
B	B					16
C	C					16
D	D	D	D			0



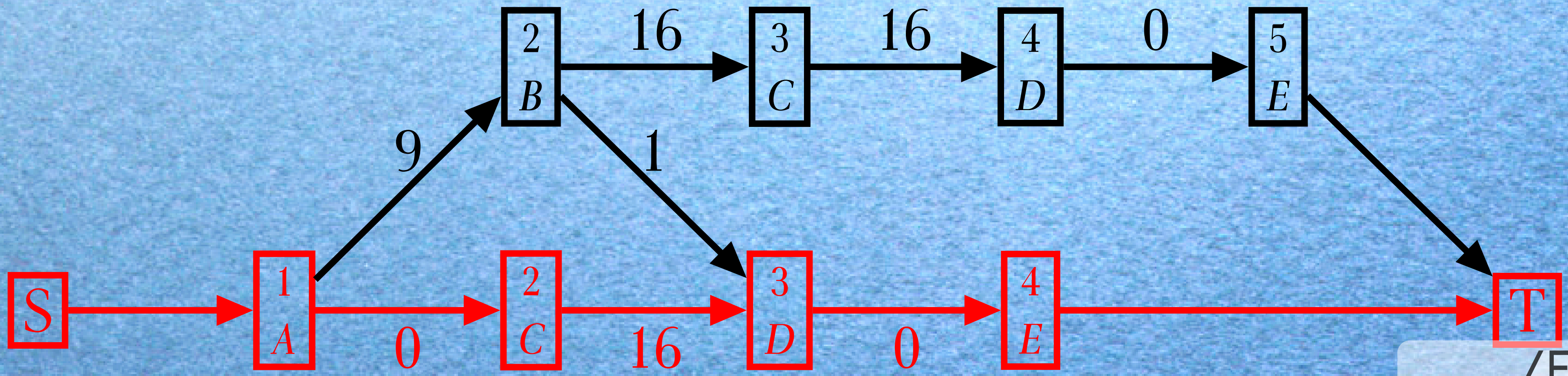
A A A	B B	C C	D D D D D

A	A	A				9
B	B	␣	C	C		1
D	D	D	D	D		0



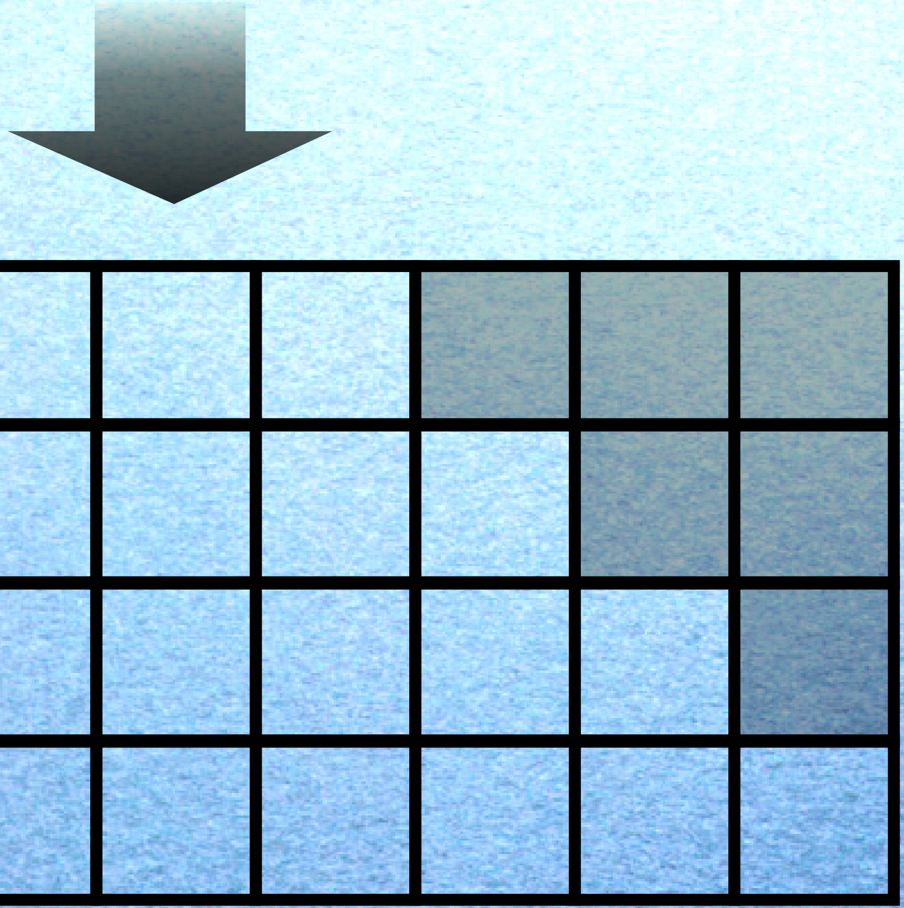
A A A	B B	C C	D D D D D

A	A	A	␣	B	B	0
C	C					16
D	D	D	D	D		0



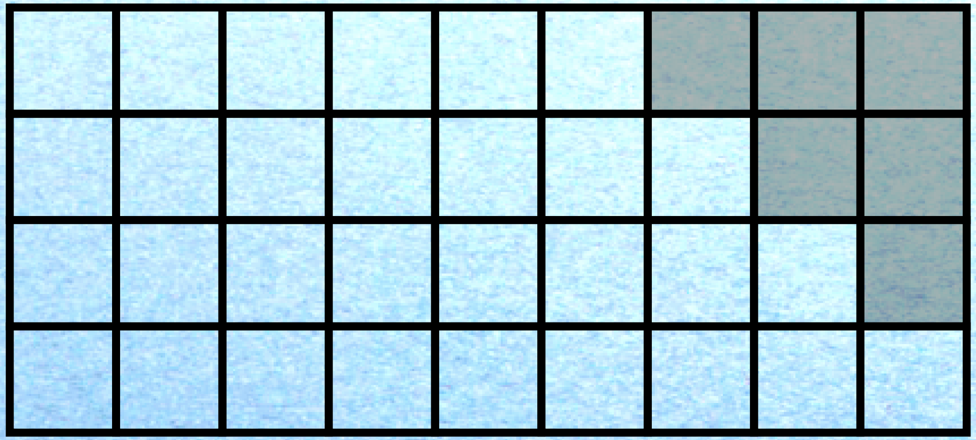
algorithm → example 1 → naïve solution

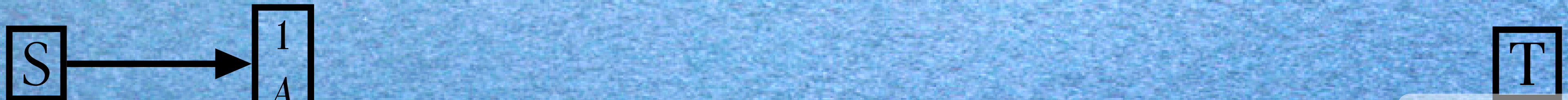
A A A | B B | C C | D D D D D



						▨	▨	▨
							▨	▨
								▨

A A A | B B | C C | D D D D D

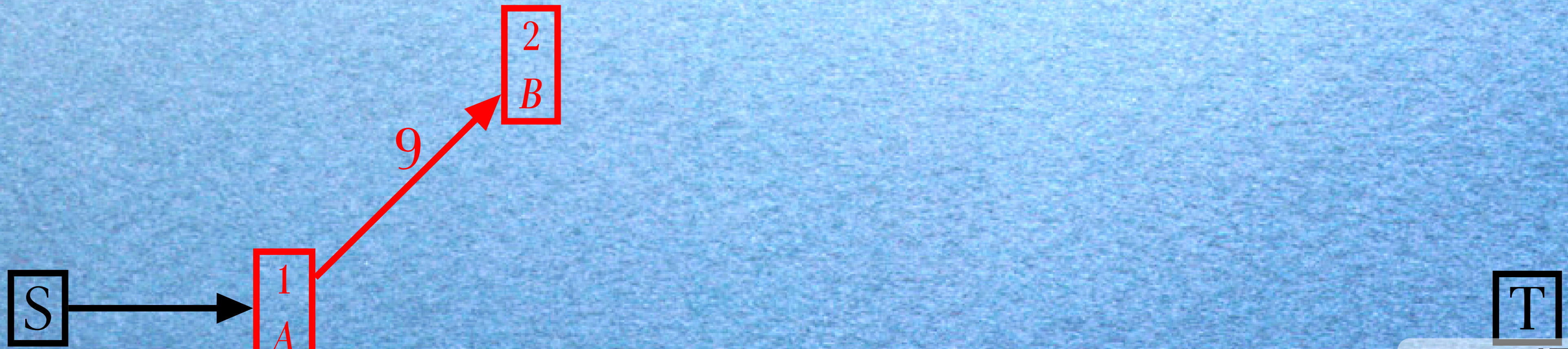




A A A | B B | C C | D D D D D

A A A						▨	▨	▨
							▨	▨
								▨

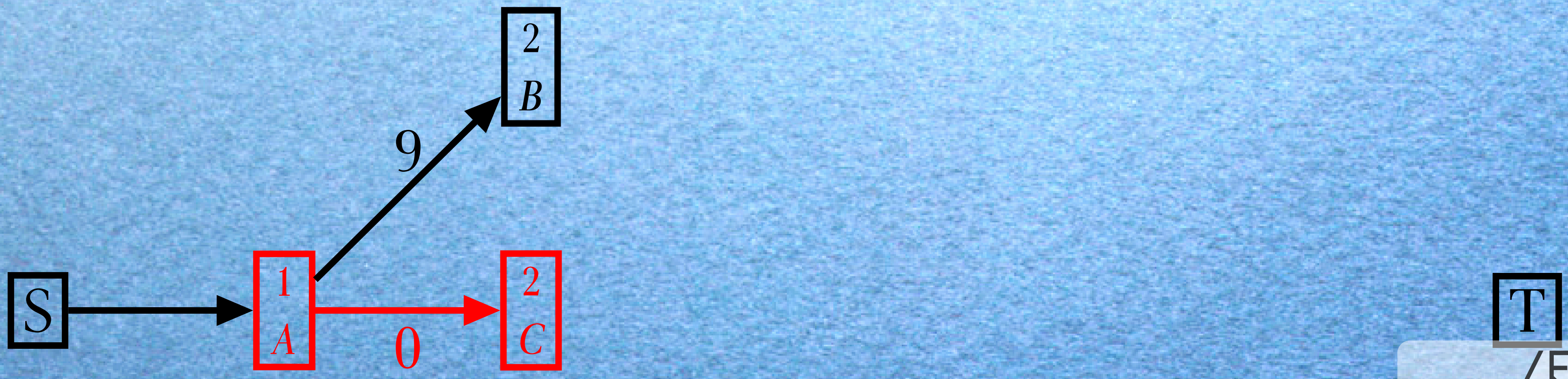
9



A A A | B B | C C | D D D D D

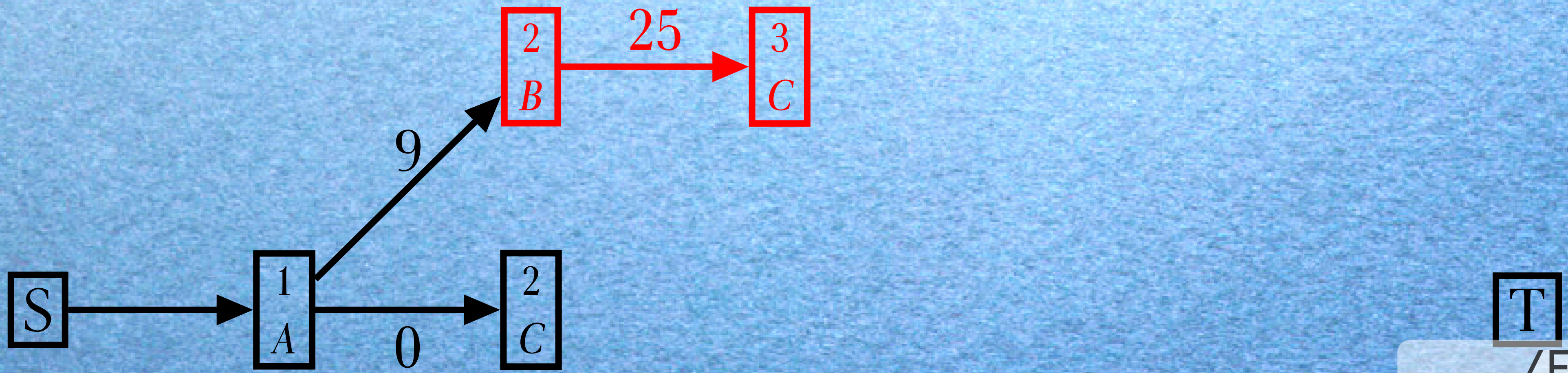
A	A	A	␣	B	B	▨	▨	▨
							▨	▨
								▨

0



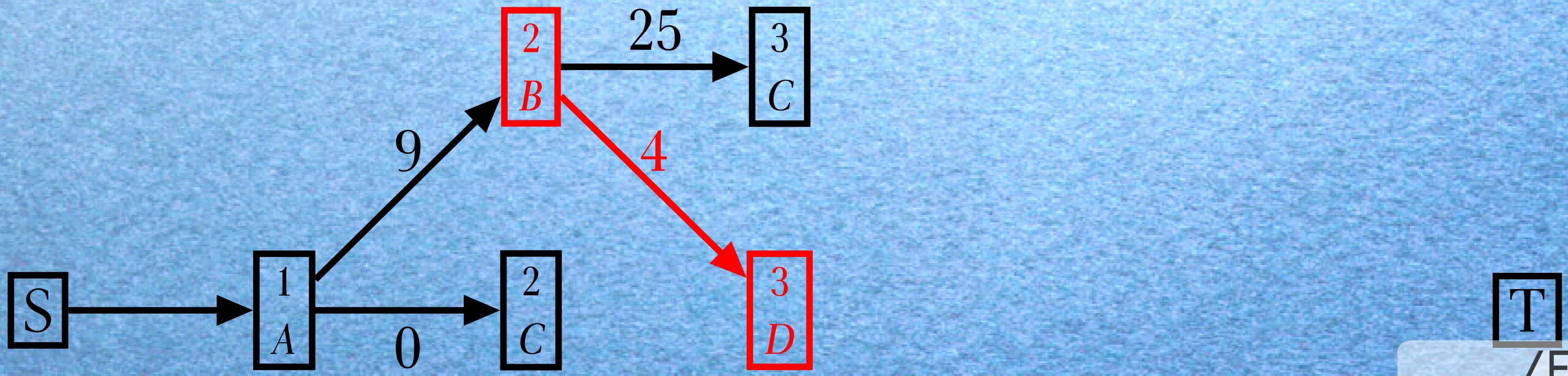
A A A | B B | C C | D D D D D

A	A	A				▨	▨	▨	9
B	B						▨	▨	25
								▨	



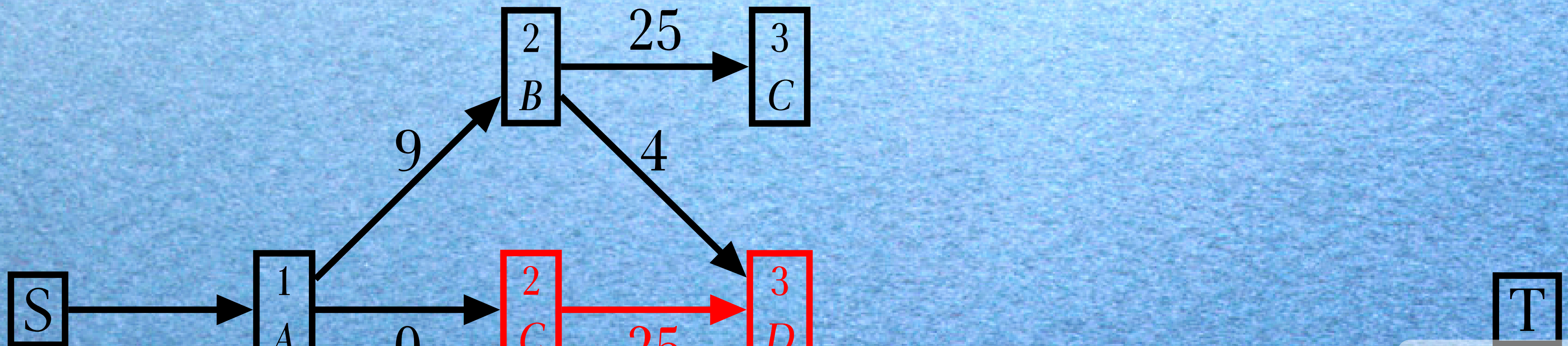
A A A | B B | C C | D D D D D

A	A	A				▨	▨	▨	9
B	B	␣	C	C			▨	▨	4
								▨	



A A A | B B | C C | D D D D D

A	A	A	␣	B	B	▨	▨	▨	0
C	C						▨	▨	25
								▨	

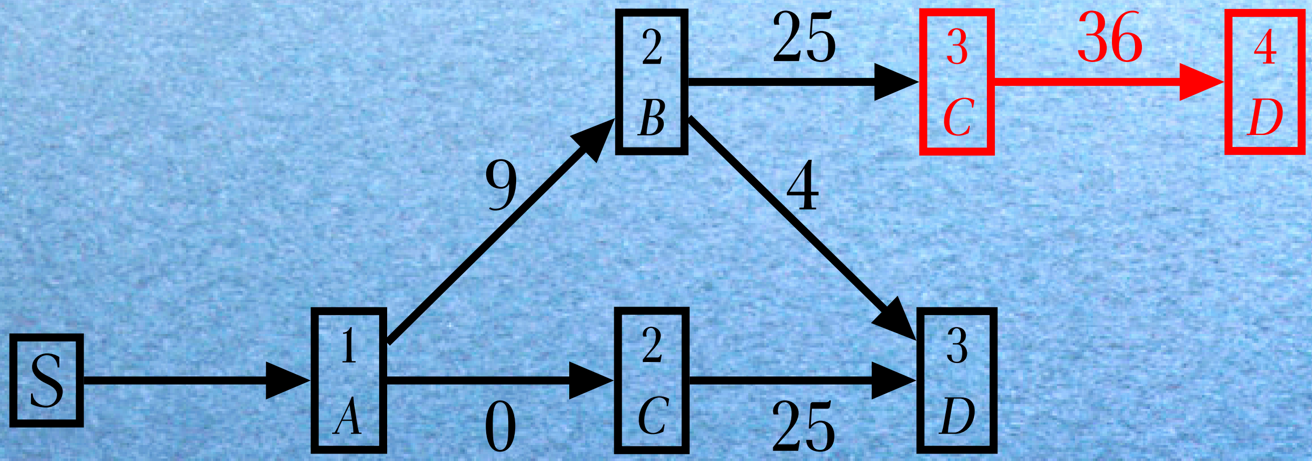


algorithm → example 2 → solution

/EURO
/BachoTEX
20\\

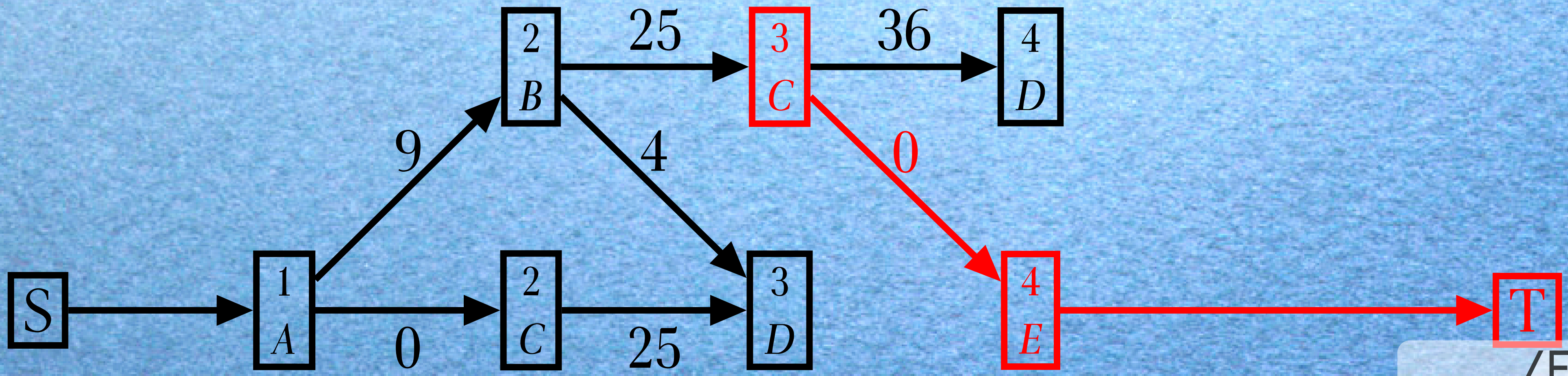
A A A | B B | C C | D D D D D

A	A	A				▨	▨	▨	9
B	B						▨	▨	25
C	C							▨	36



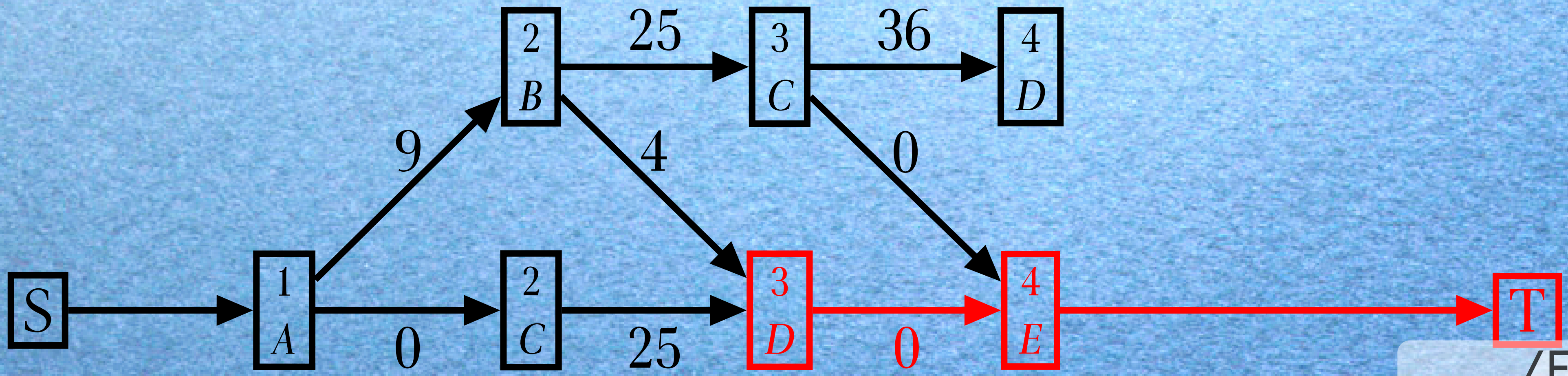
A A A | B B | C C | D D D D D

A	A	A				▨	▨	▨	9
B	B						▨	▨	25
C	C	␣	D	D	D	D	D	▨	0



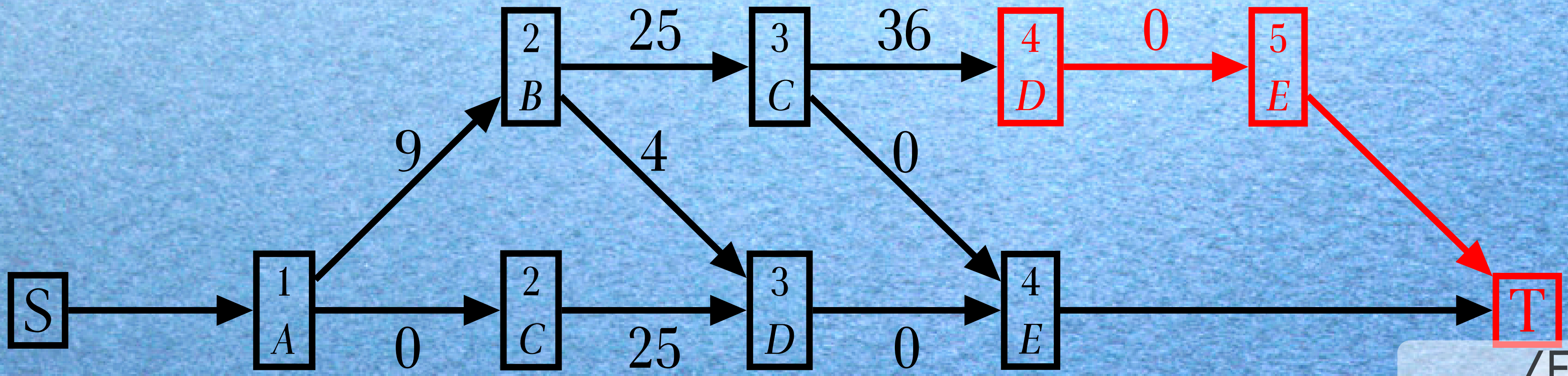
A A A | B B | C C | D D D D D

A	A	A				▨	▨	▨	9
B	B	␣	C	C			▨	▨	4
D	D	D	D	D				▨	0



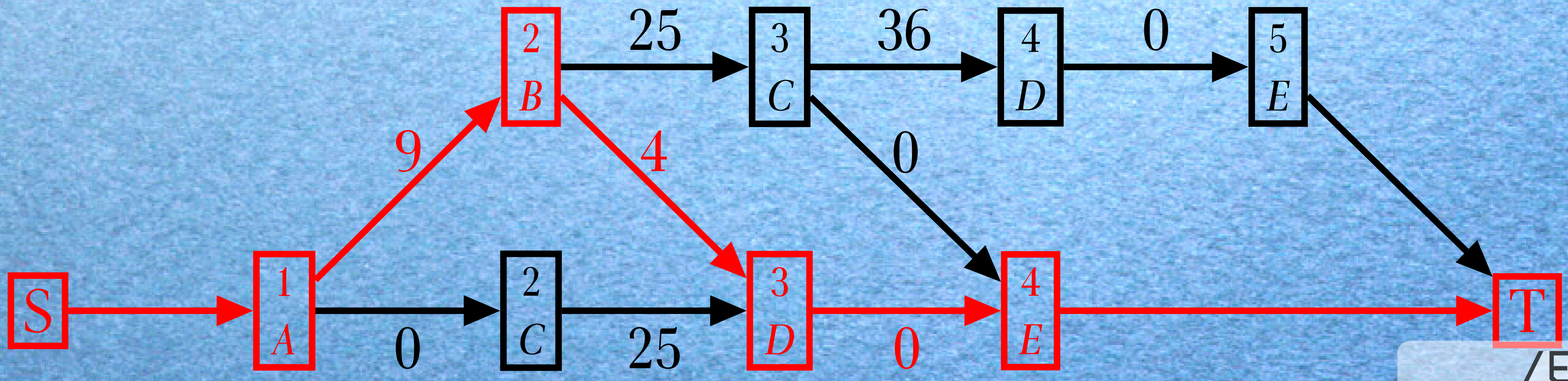
A A A | B B | C C | D D D D D

A	A	A				▨	▨	▨	9
B	B						▨	▨	25
C	C							▨	36
D	D	D	D						0



A A A | B B | C C | D D D D D

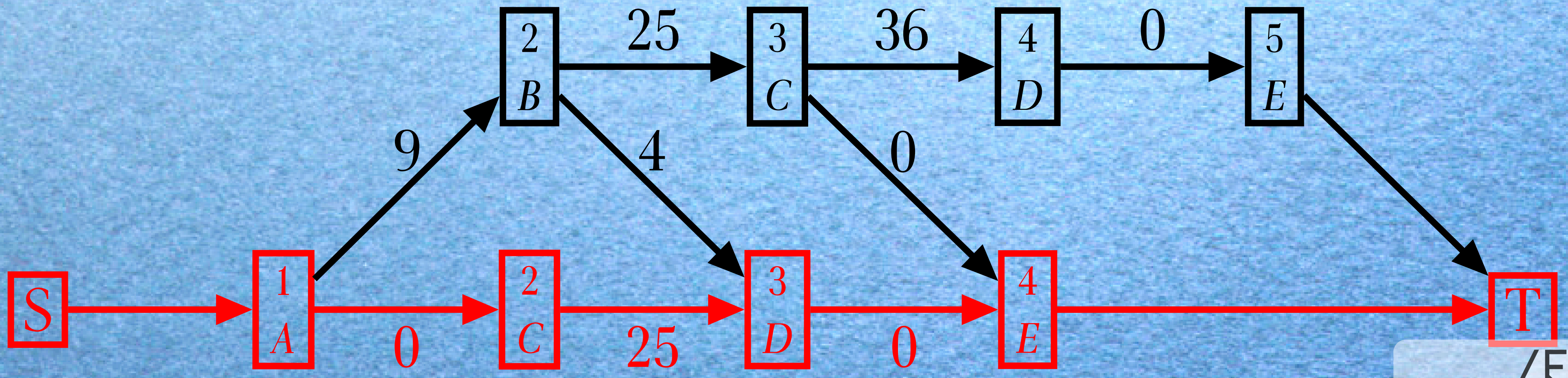
A	A	A				▨	▨	▨	9
B	B	␣	C	C			▨	▨	4
D	D	D	D	D				▨	0

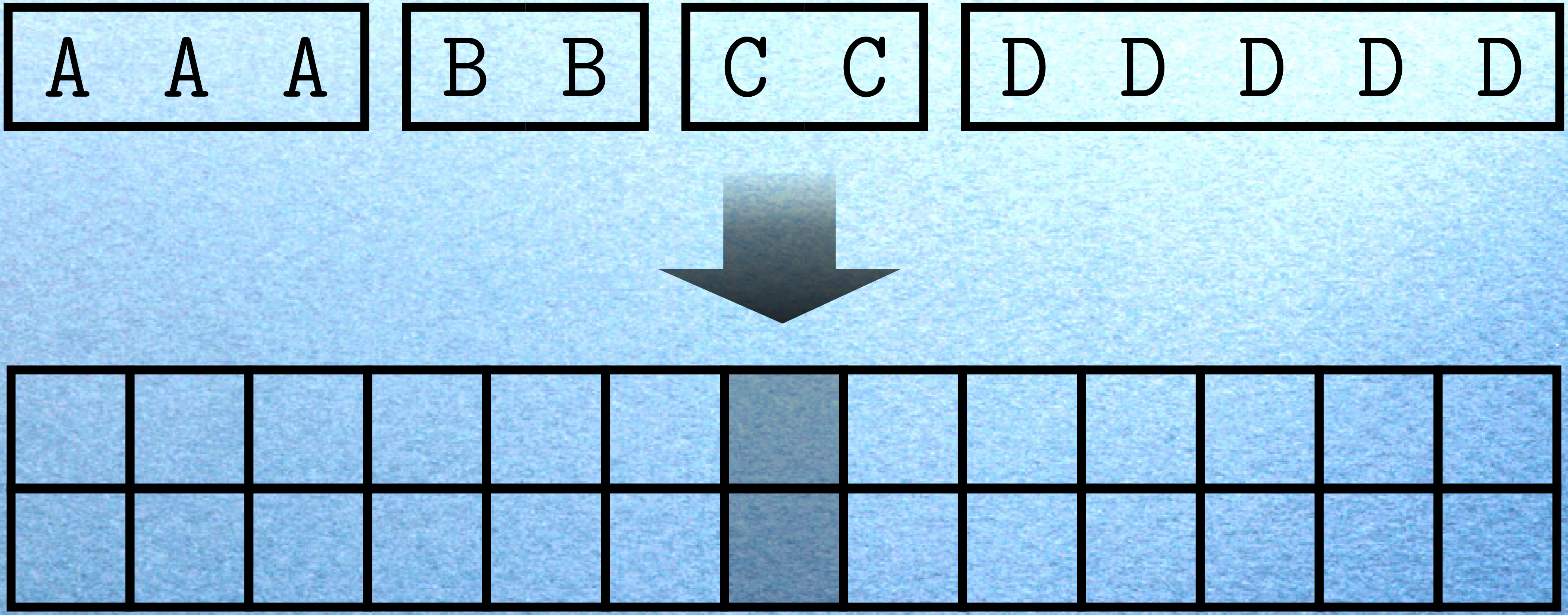


algorithm → example 2 → optimal solution

A A A	B B	C C	D D D D D

A	A	A	␣	B	B	▨	▨	▨	0
C	C						▨	▨	25
D	D	D	D	D				▨	0





A A A

B B

C C

D D D D D

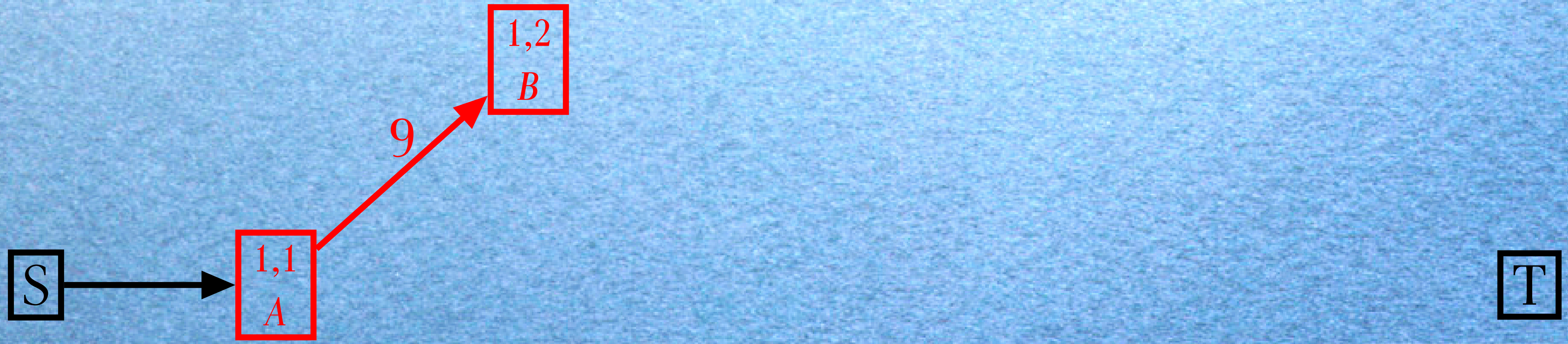
A A A | B B | C C | D D D D D

						▨						
						▨						

A A A　B B　C C　D D D D D

A	A	A				▨						
						▨						

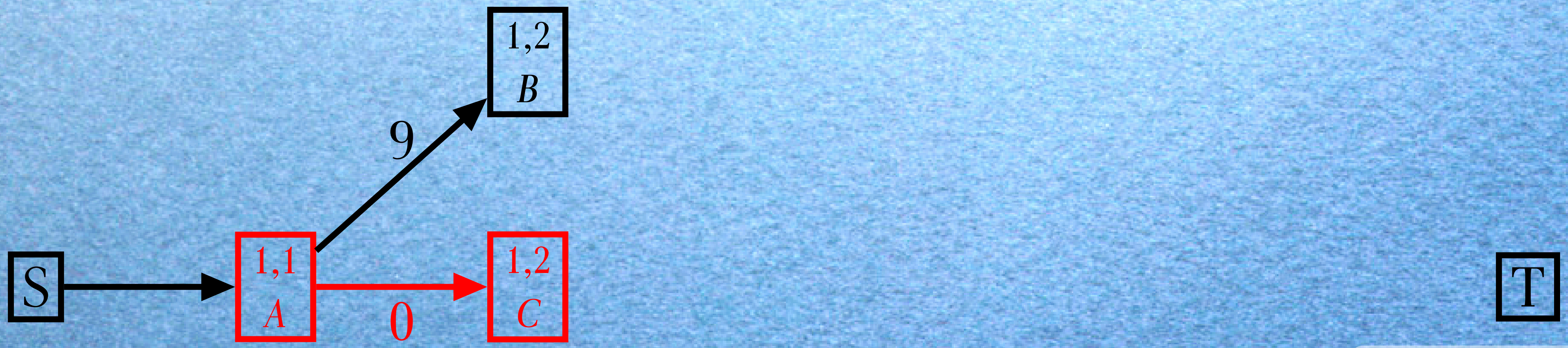
9



A A A | B B | C C | D D D D D

A	A	A	␣	B	B	▨						
						▨						

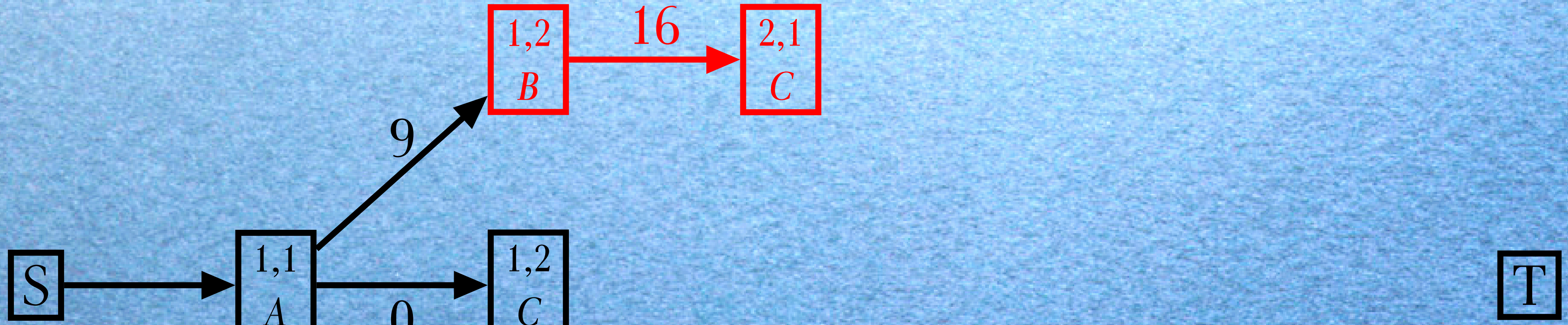
0



A A A | B B | C C | D D D D D

A	A	A				▨	B	B				
						▨						

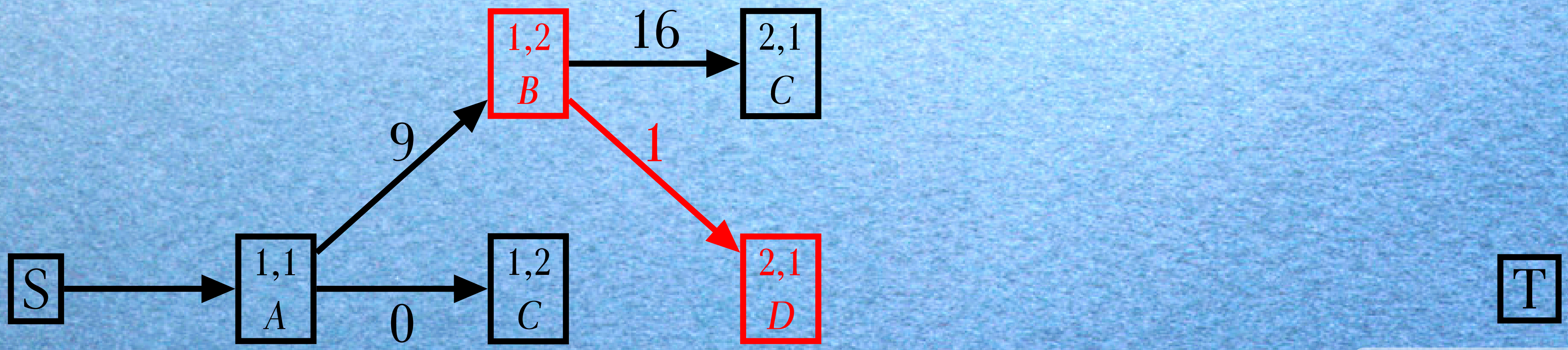
$9 + 16$



A A A | B B | C C | D D D D D

A A A						▨	B B ␣			C C		
						▨						

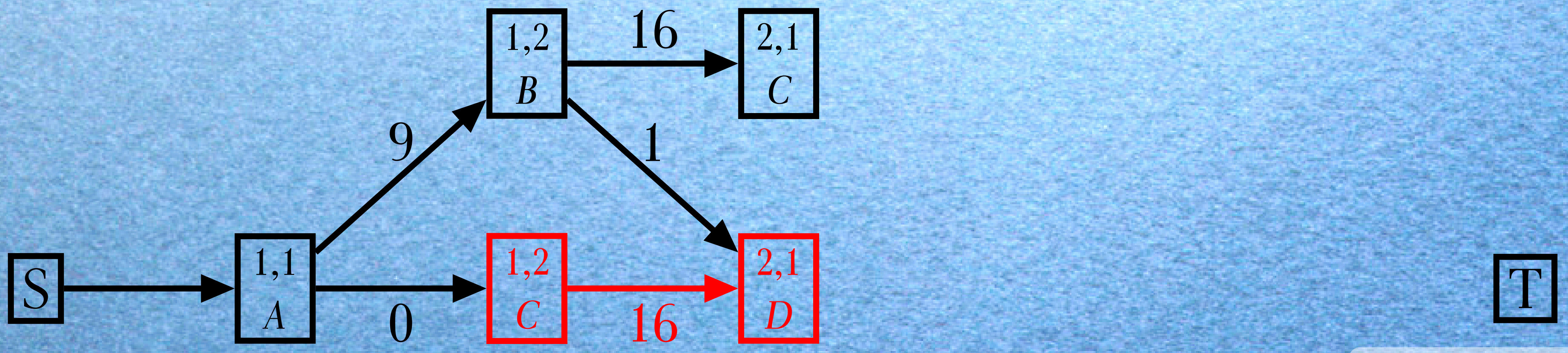
$9 + 1$



A	A	A	B	B	C	C	D	D	D	D	D	

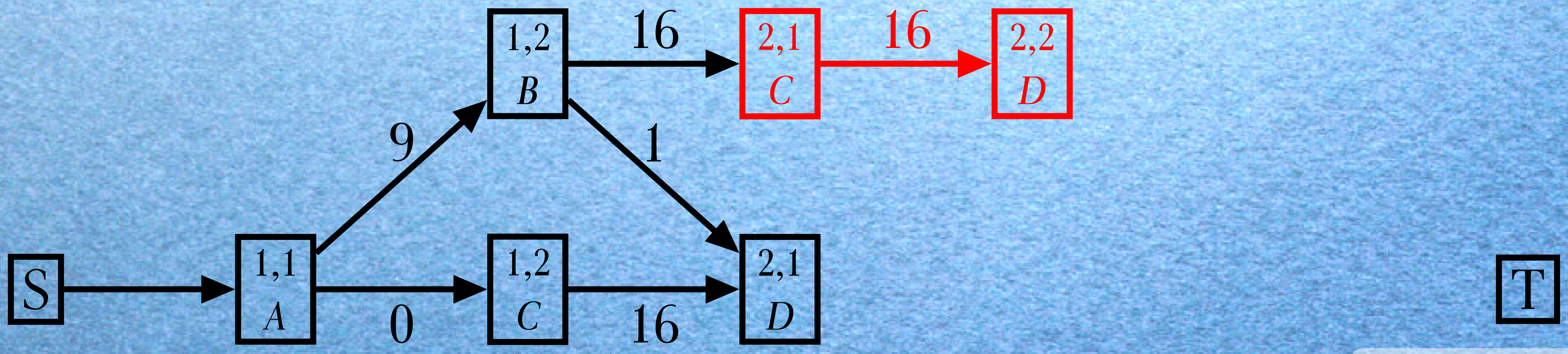
A	A	A	␣	B	B	▨	C	C				
						▨						

$0 + 16$



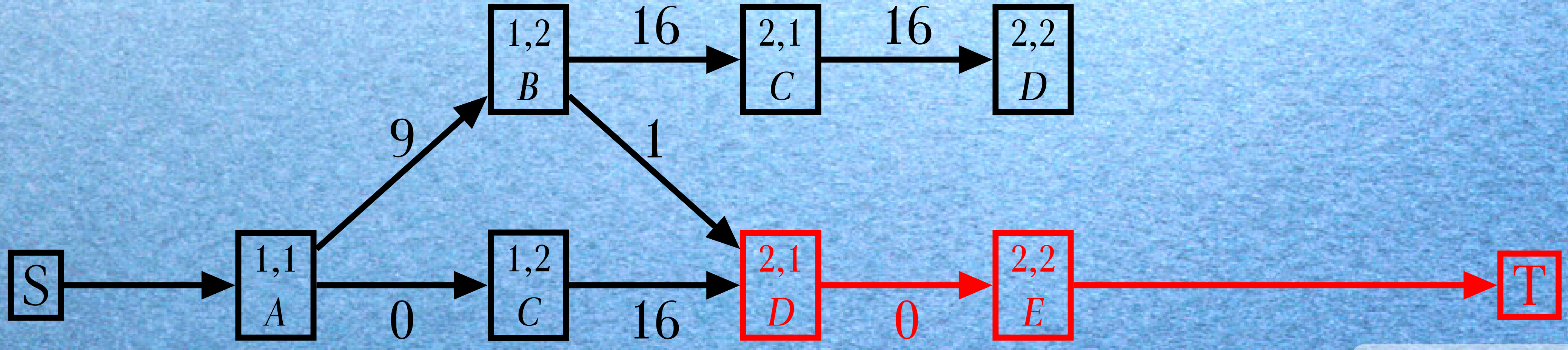
A A A	B B	C C	D D D D D

A A A						▨	B B						$9 + 16$
C C						▨							16



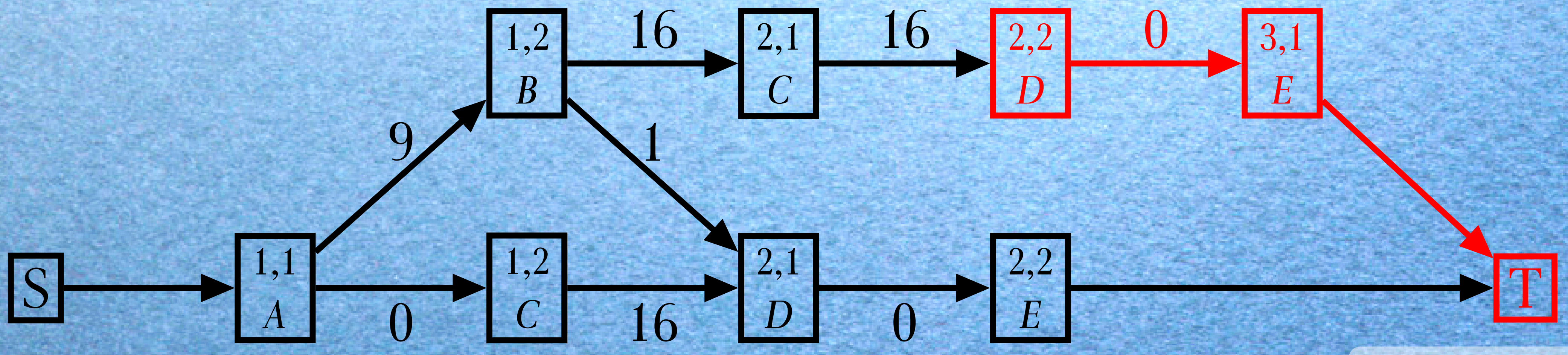
A A A | B B | C C | D D D D D

A	A	A				▨	B	B	␣	C	C		9 + 1
D	D	D	D	D		▨							0



A A A	B B	C C	D D D D D

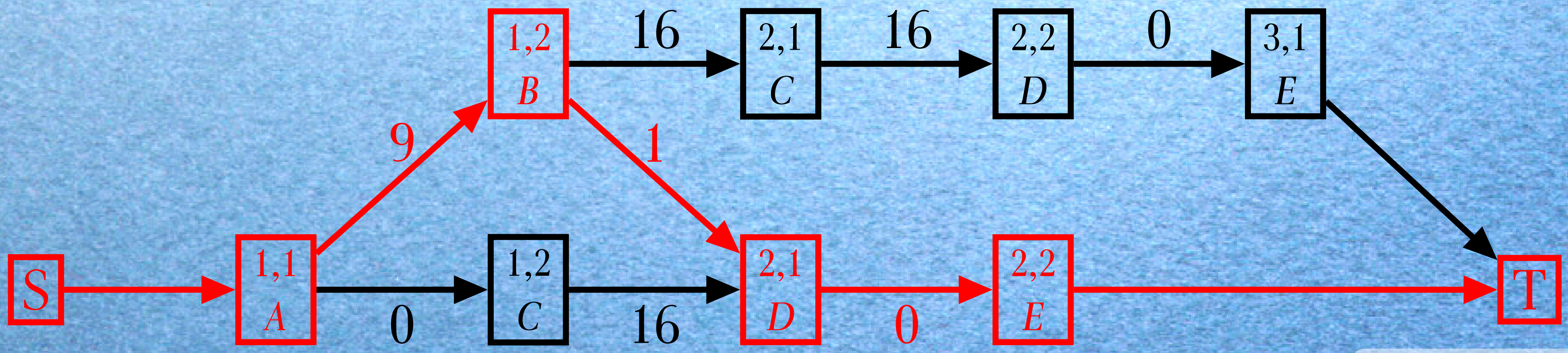
<table>
<tr><td colspan="3">A A A</td><td></td><td></td><td></td><td>▨</td><td colspan="2">B B</td><td></td><td></td><td></td><td></td><td>9 + 16</td></tr>
<tr><td colspan="2">C C</td><td></td><td></td><td></td><td></td><td>▨</td><td colspan="4">D D D D</td><td></td><td></td><td>16 + 0</td></tr>
</table>



algorithm → example 3 → solution

A A A | B B | C C | D D D D D

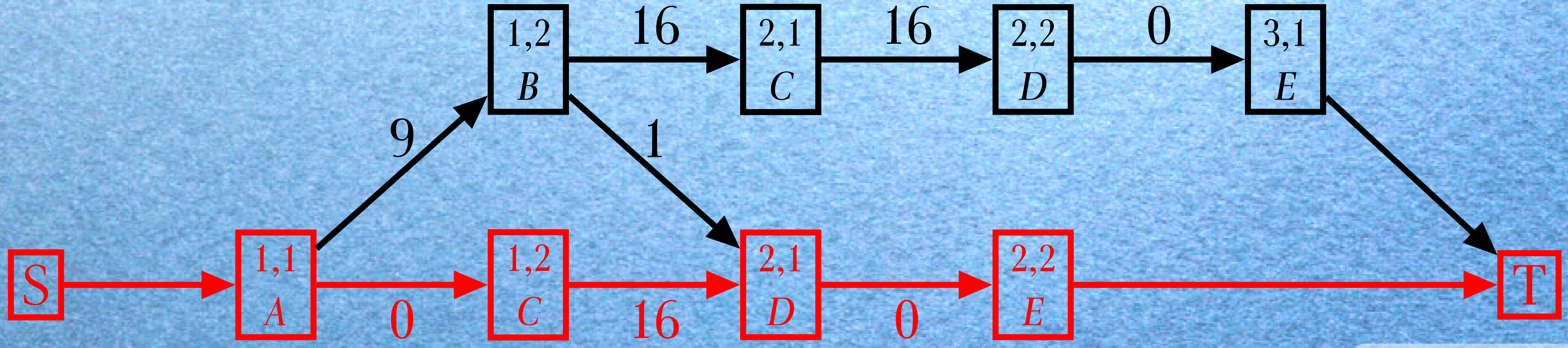
A	A	A				▨	B	B	␣	C	C		$9+1$
D	D	D	D	D		▨							0



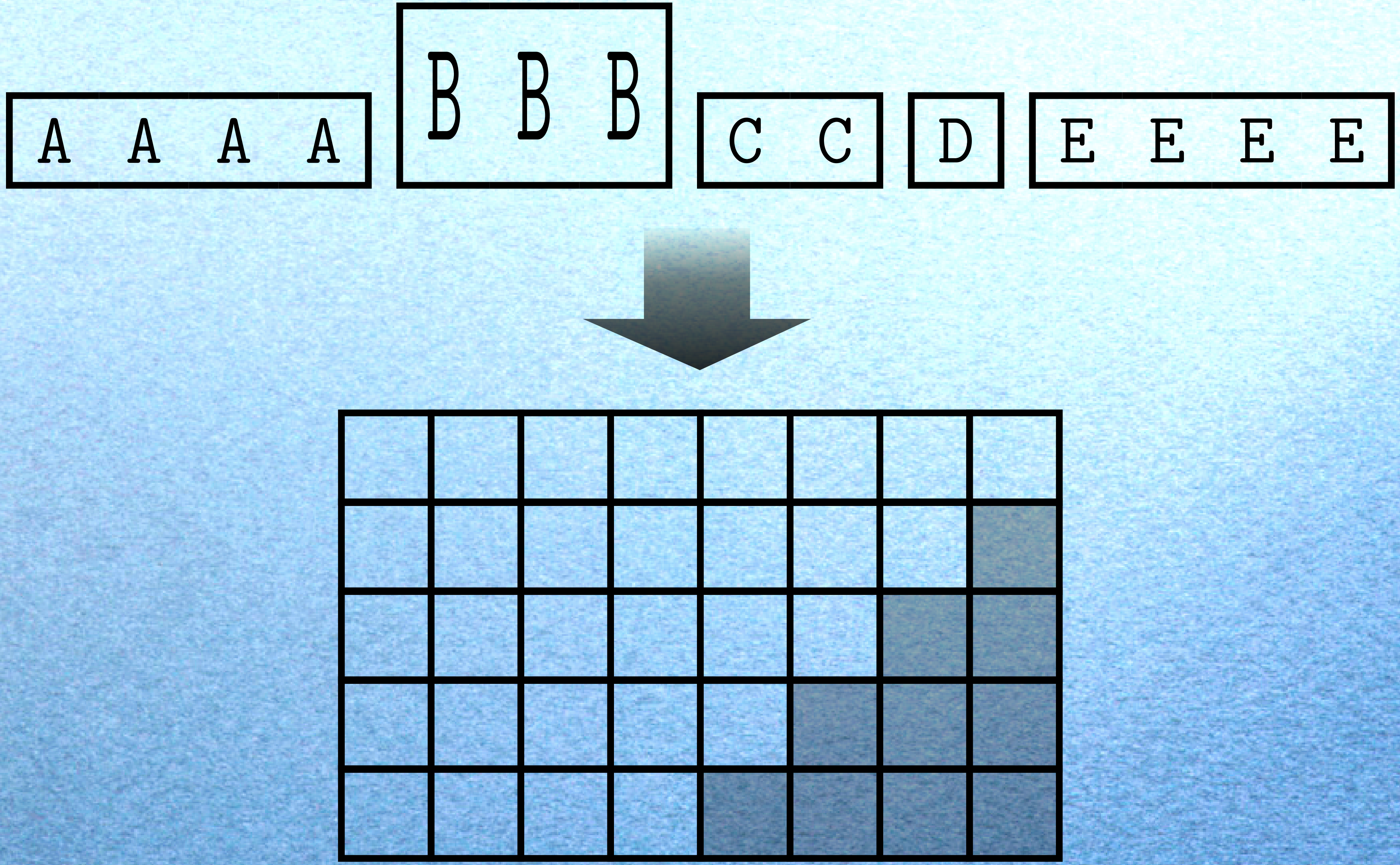
algorithm → example 3 → optimal solution

A A A	B B	C C	D D D D D

A	A	A	␣	B	B	▨	C	C					0 + 16
D	D	D	D	D		▨							0



algorithm → example 3 → naïve solution



A A A A

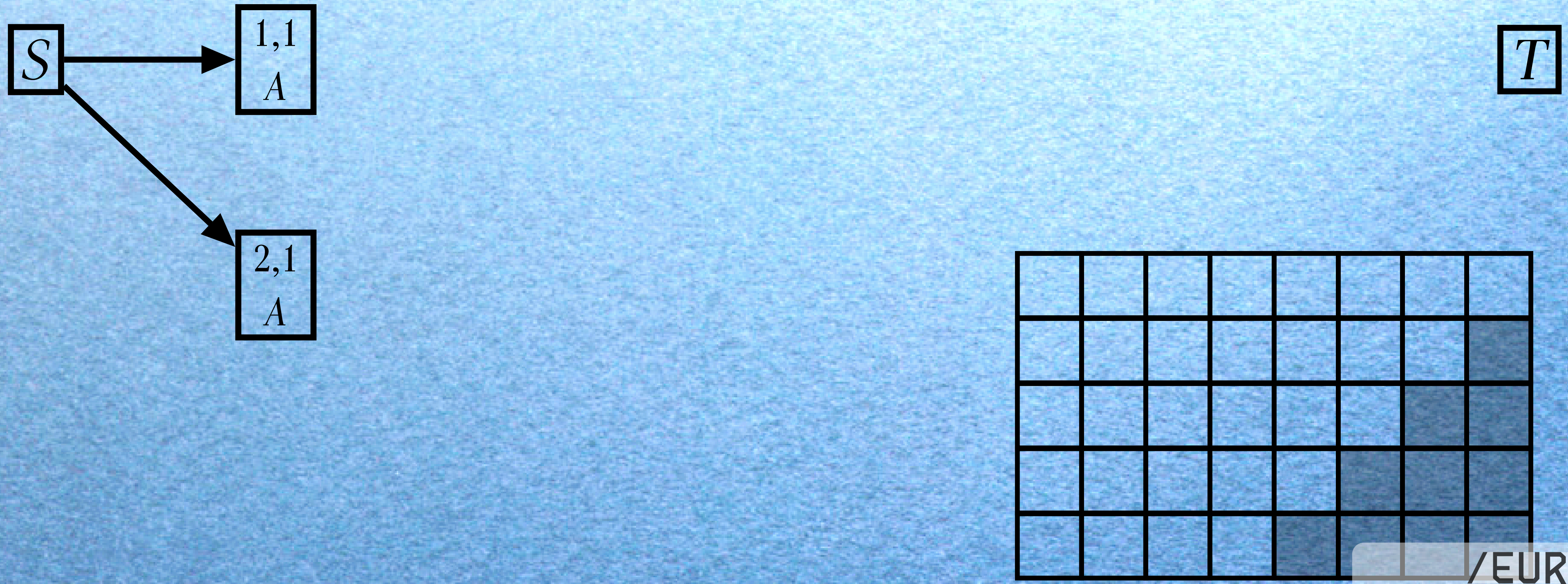
B B B

C C

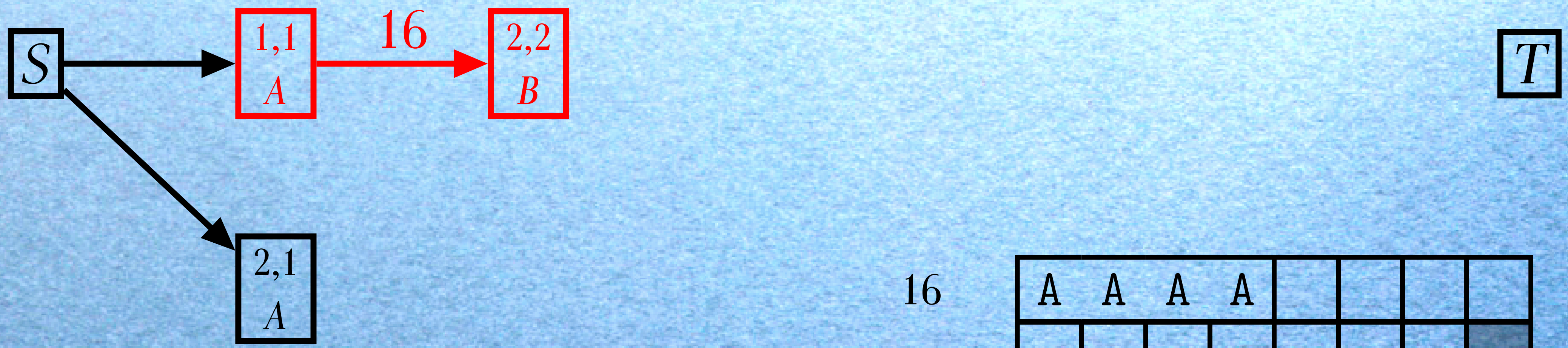
D

E E E E

A A A A B B B C C D E E E E



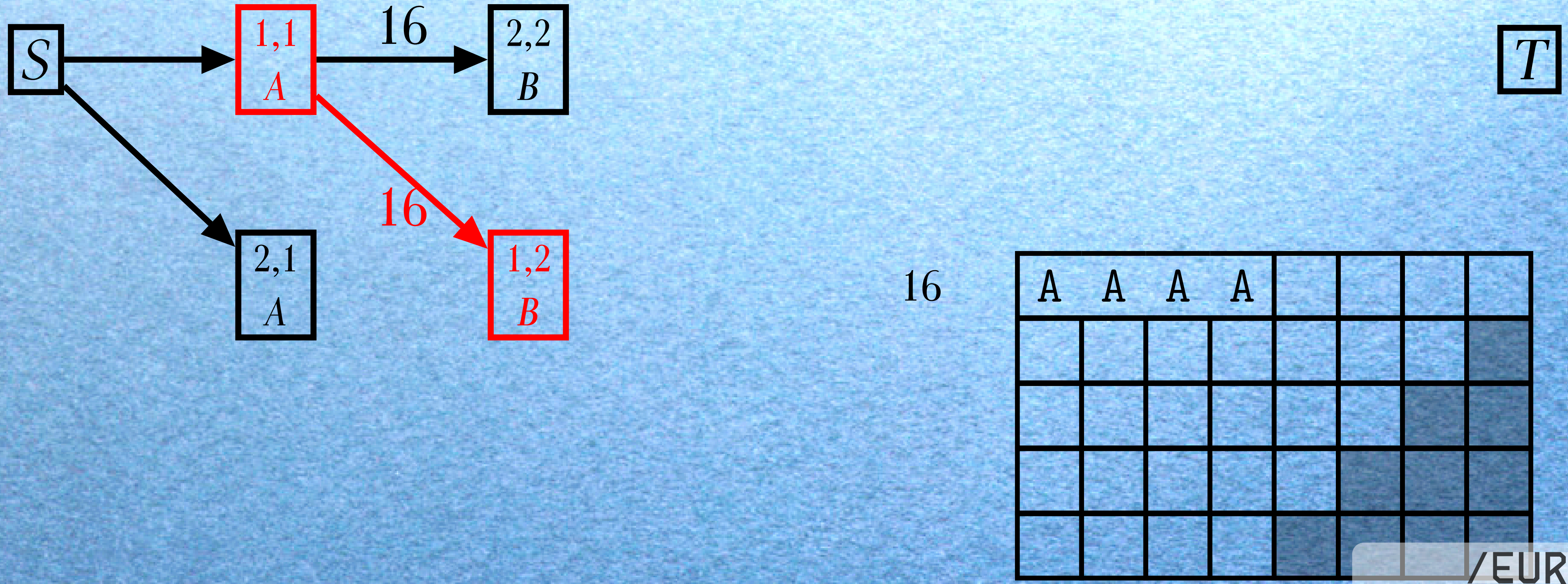
A A A A | B B B | C C | D | E E E E

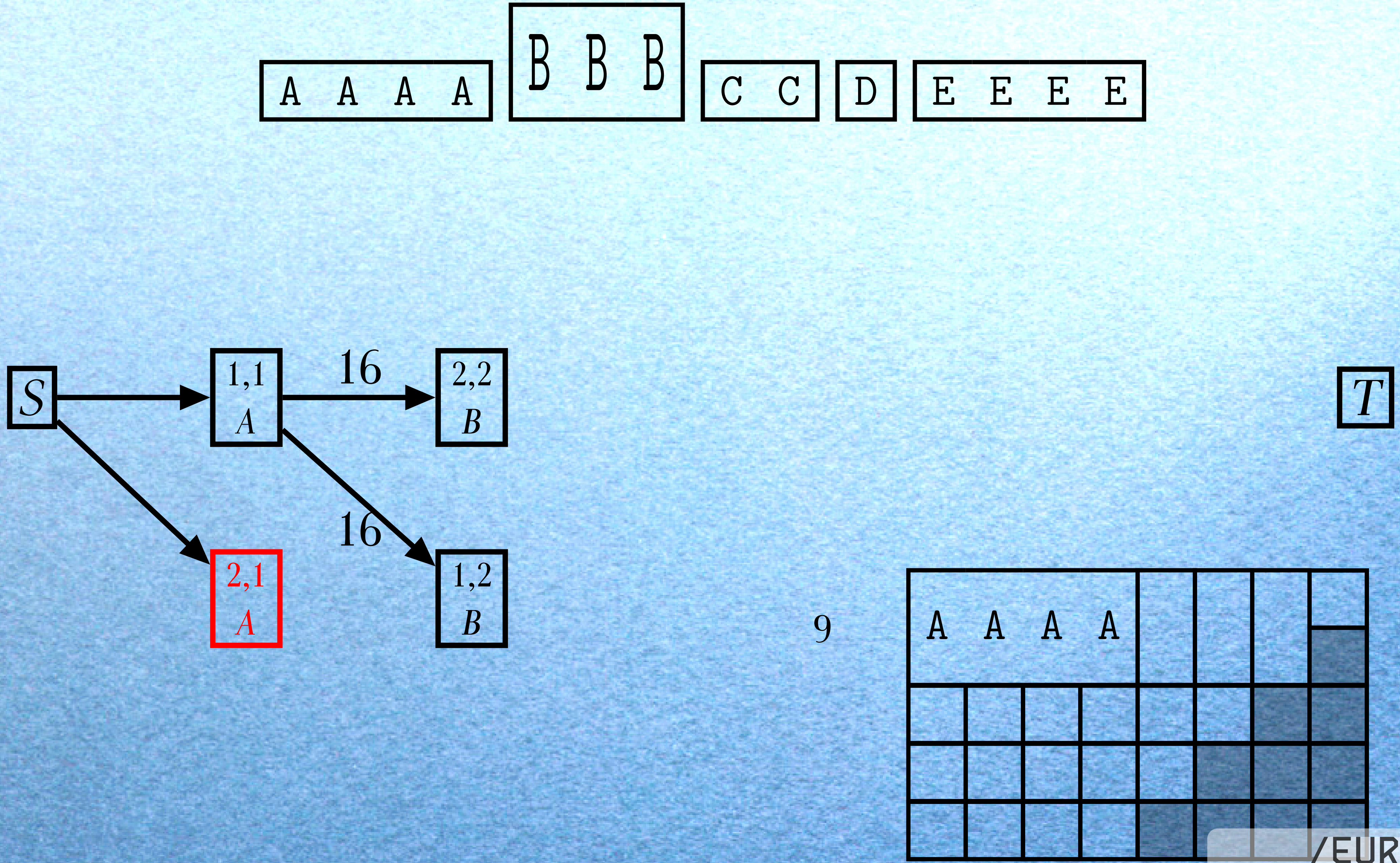


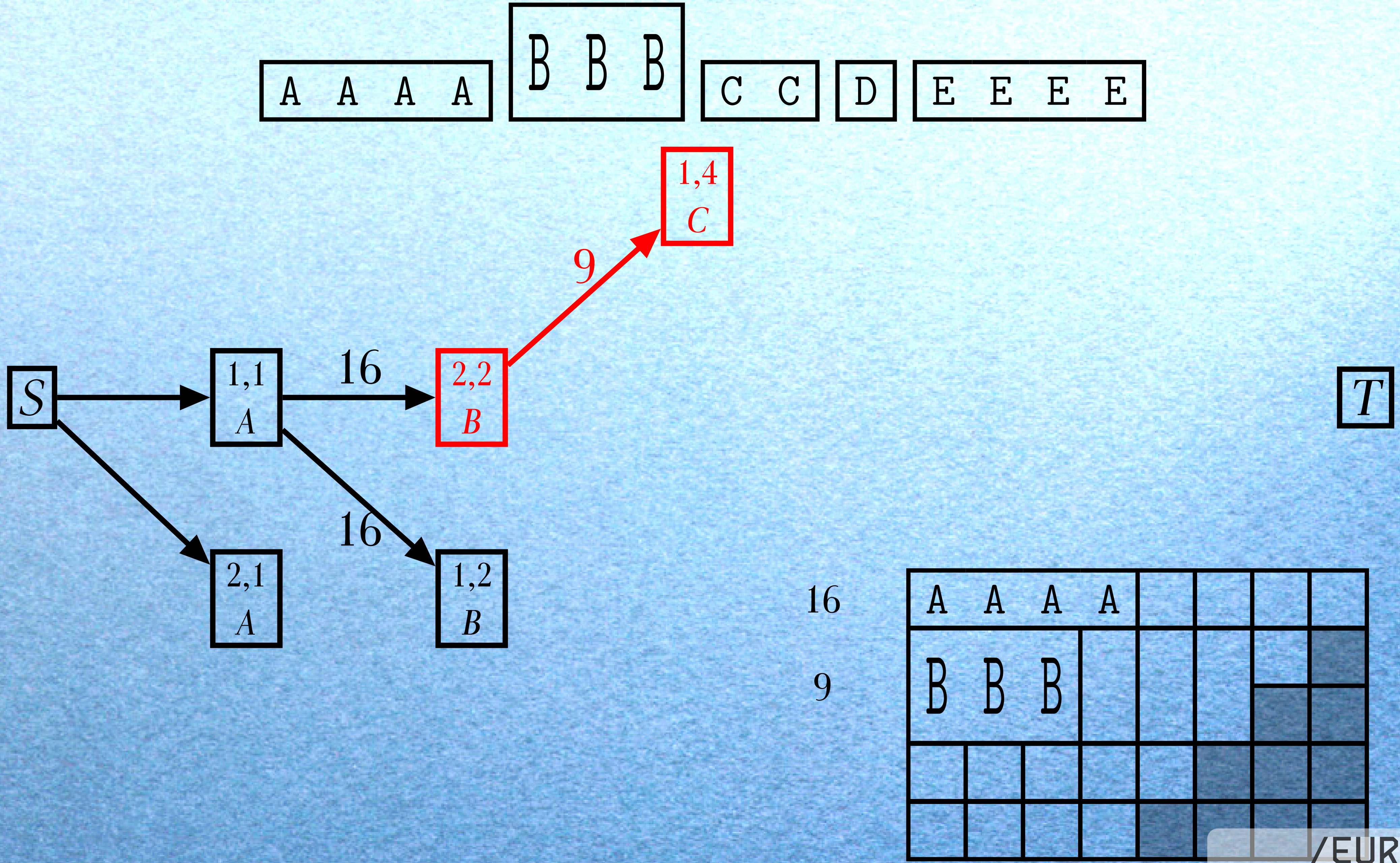
16

A	A	A	A				
							▨
						▨	▨
					▨	▨	▨
				▨	▨	▨	▨

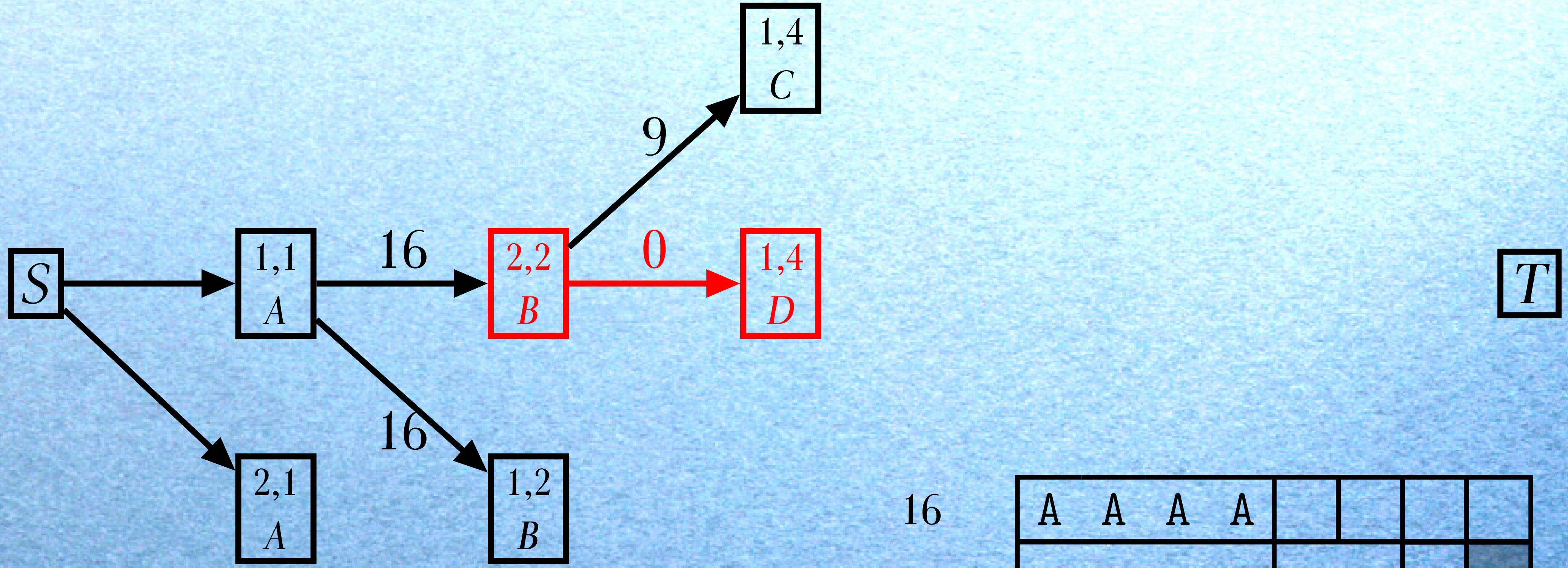
A A A A   B B B   C C   D   E E E E

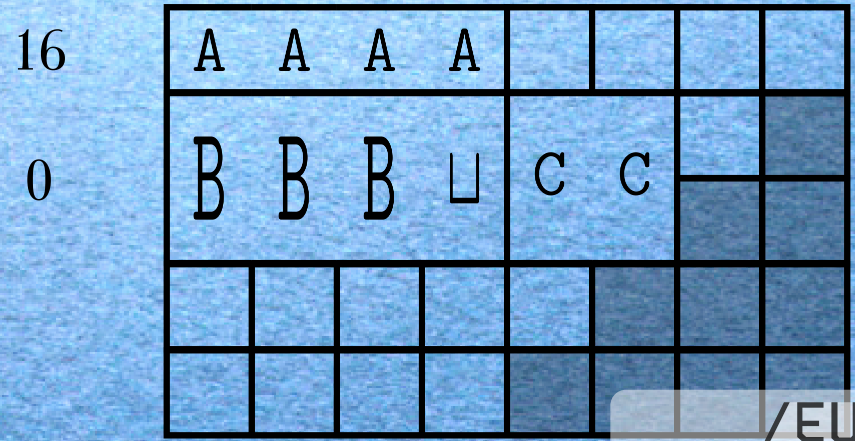



A A A A
B B B
C C
D
E E E E
S
1,1 A
16
2,2 B
2,1 A
16
1,2 B
T
9
A A A A

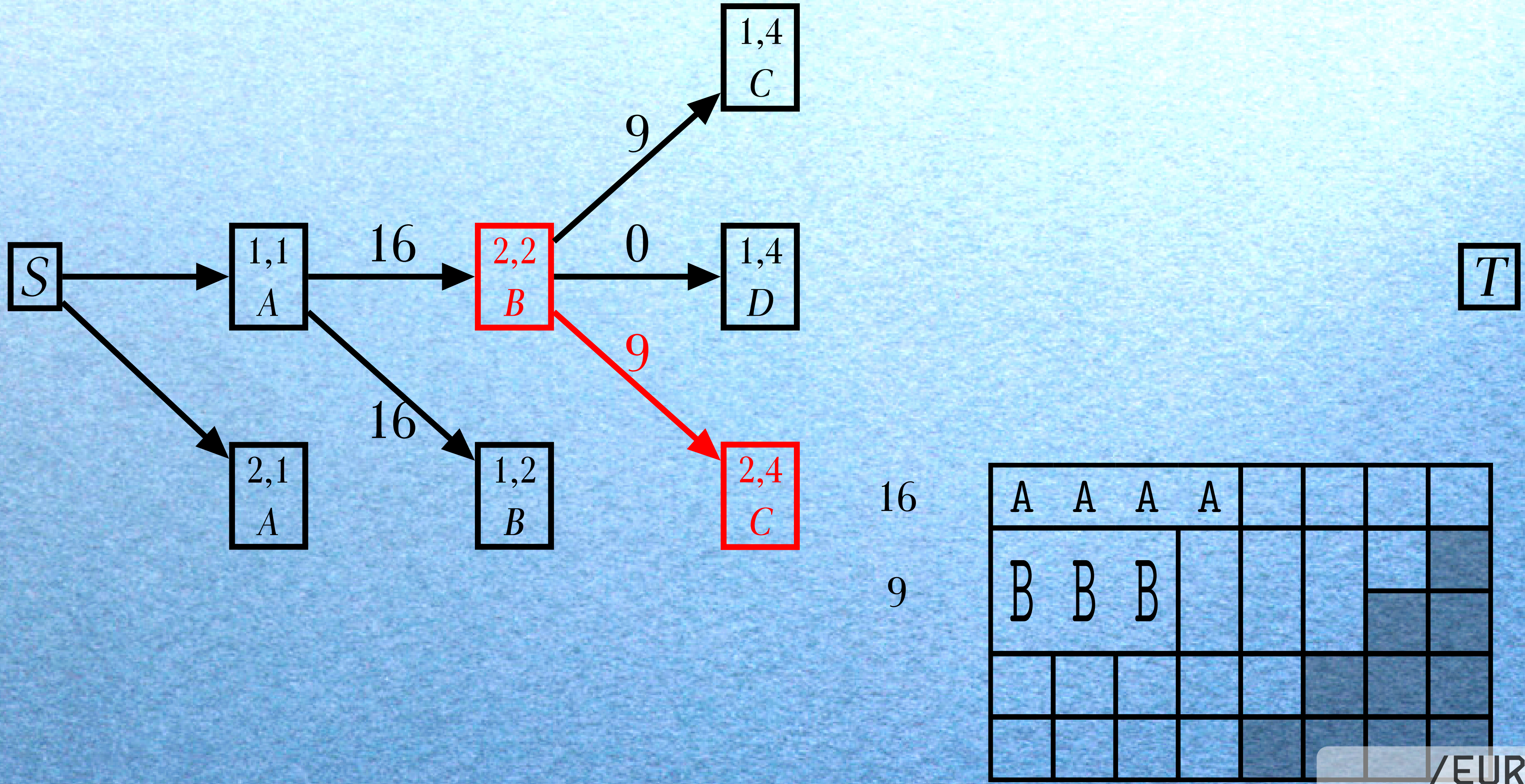

A A A A
B B B
C C
D
E E E E
1,4
C
9
S
1,1
A
16
2,2
B
T
16
2,1
A
1,2
B
16
9
A A A A
B B B

A A A A B B B C C D E E E E

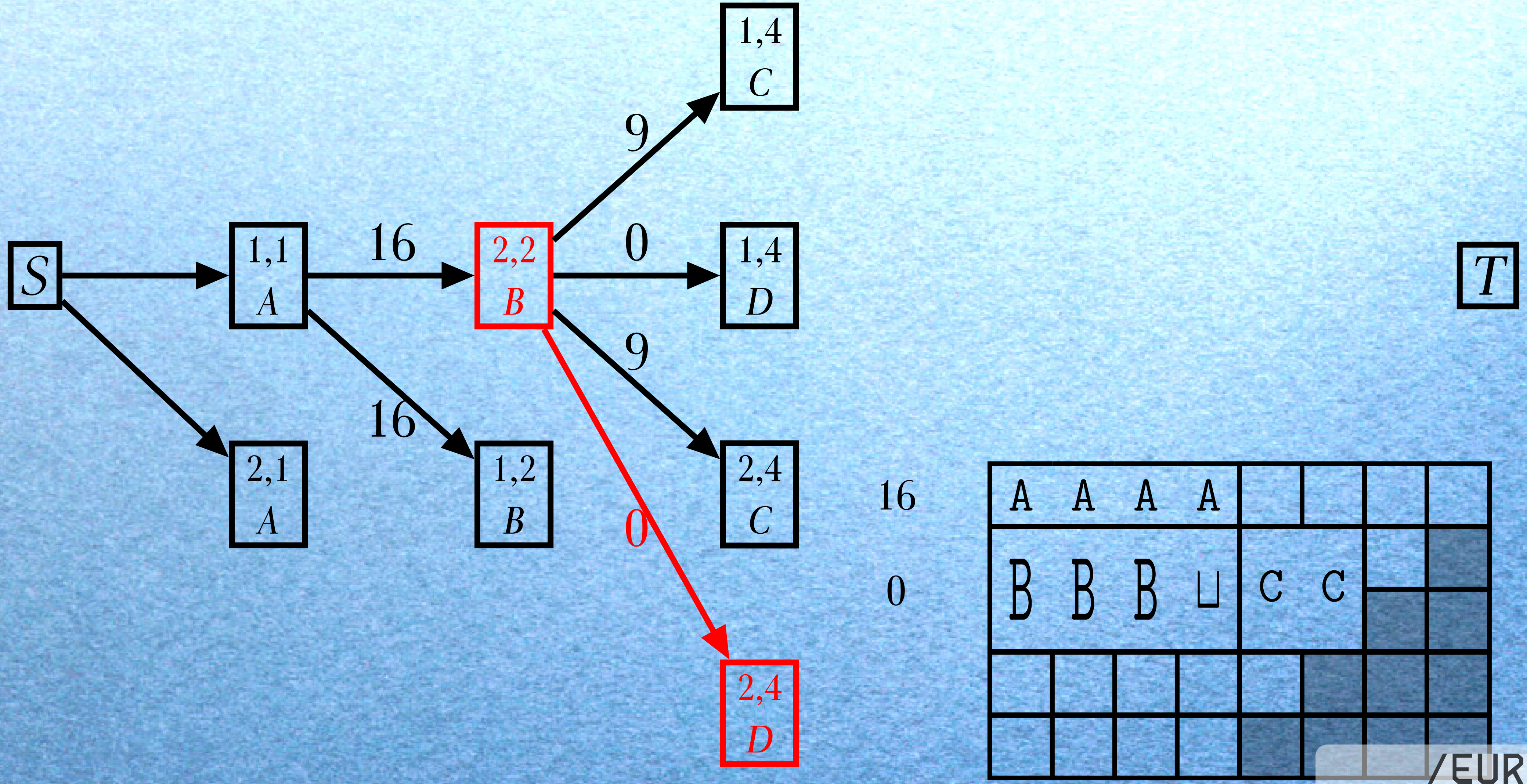


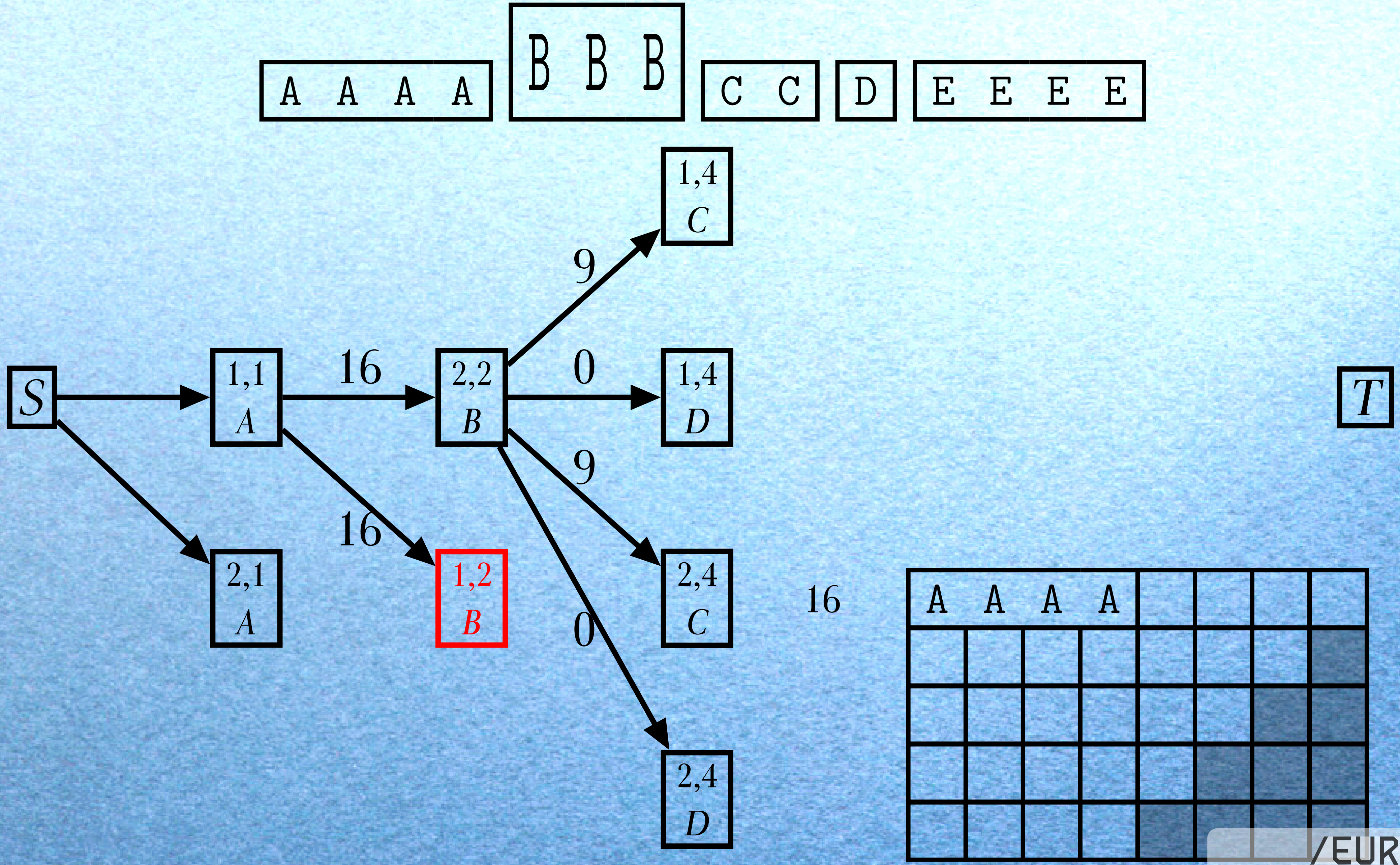


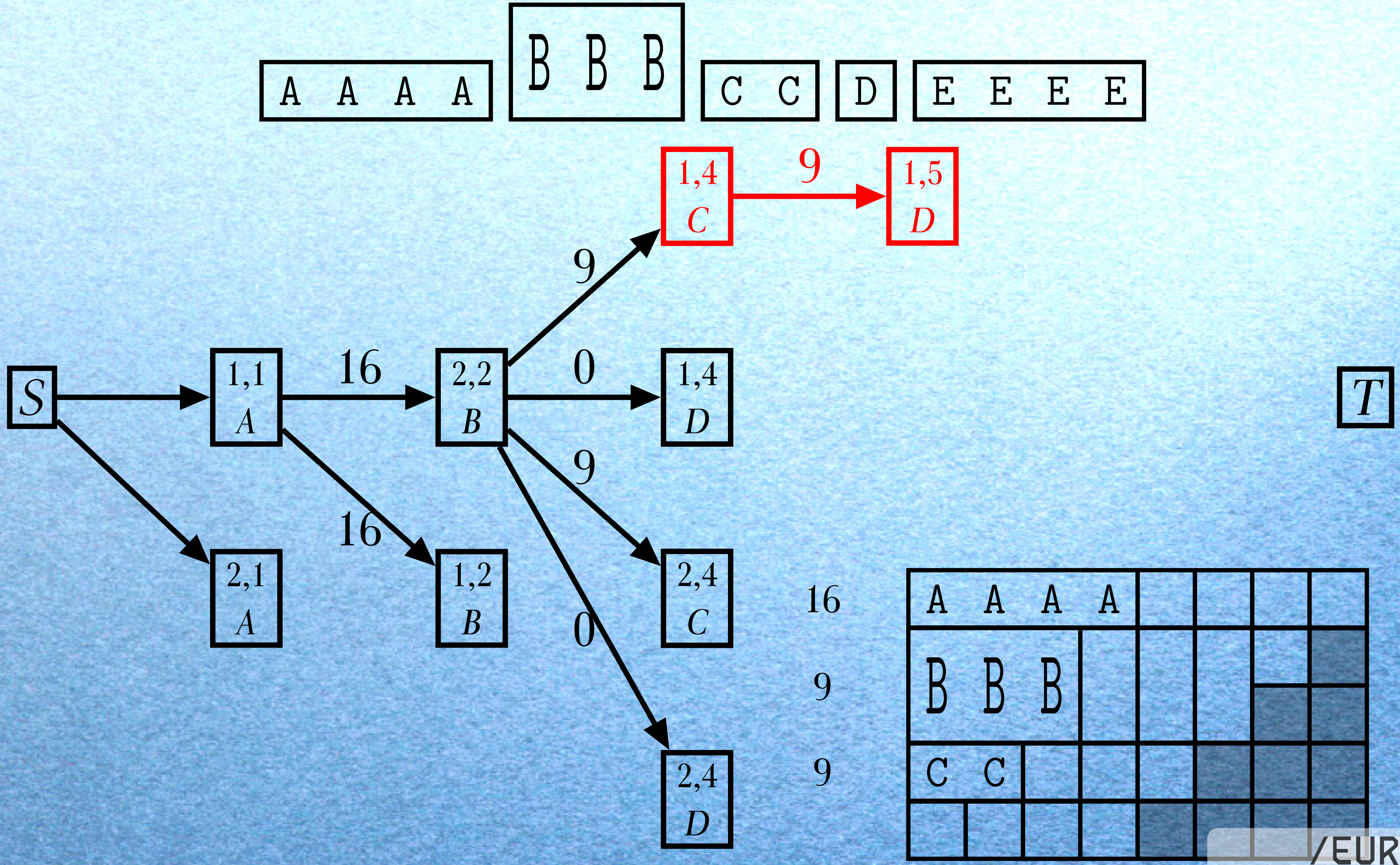
A A A A | B B B | C C | D | E E E E



A A A A B B B C C D E E E E

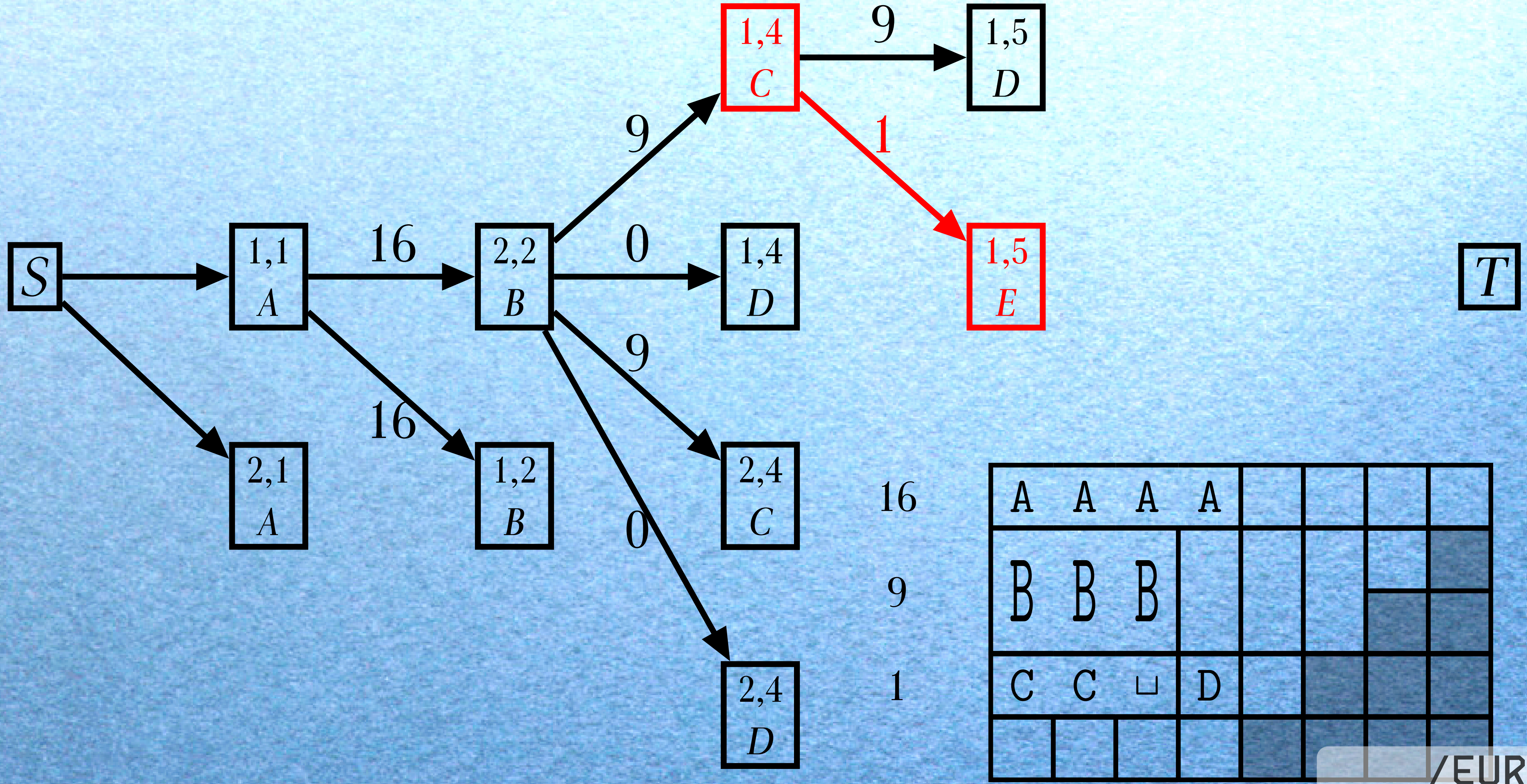



A A A A
B B B
C C
D
E E E E
S
1,1
A
2,1
A
16
16
2,2
B
1,2
B
9
0
9
0
1,4
C
1,4
D
2,4
C
2,4
D
T
16
A A A A

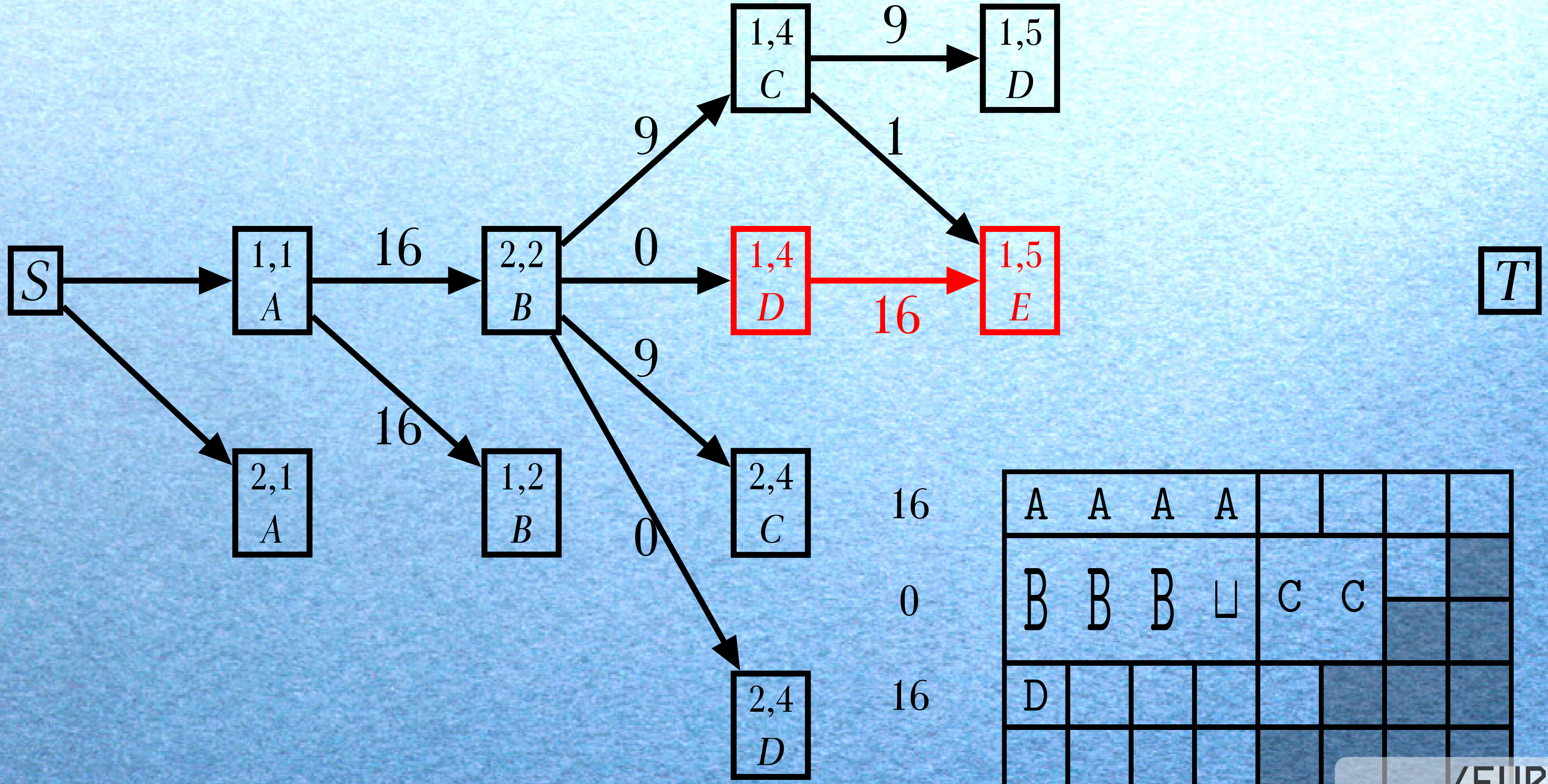


algorithm → example 4 → solution

A A A A B B B C C D E E E E

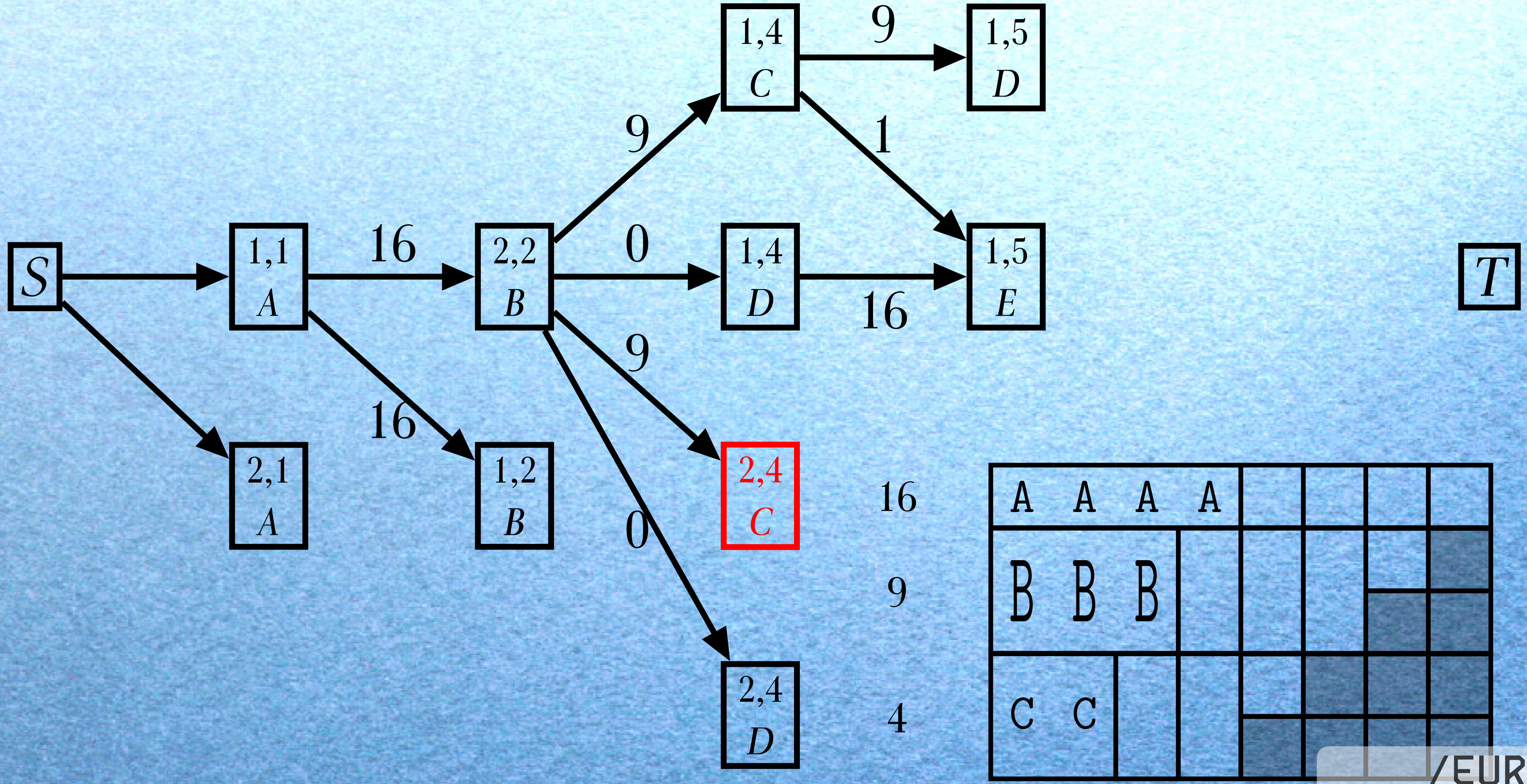


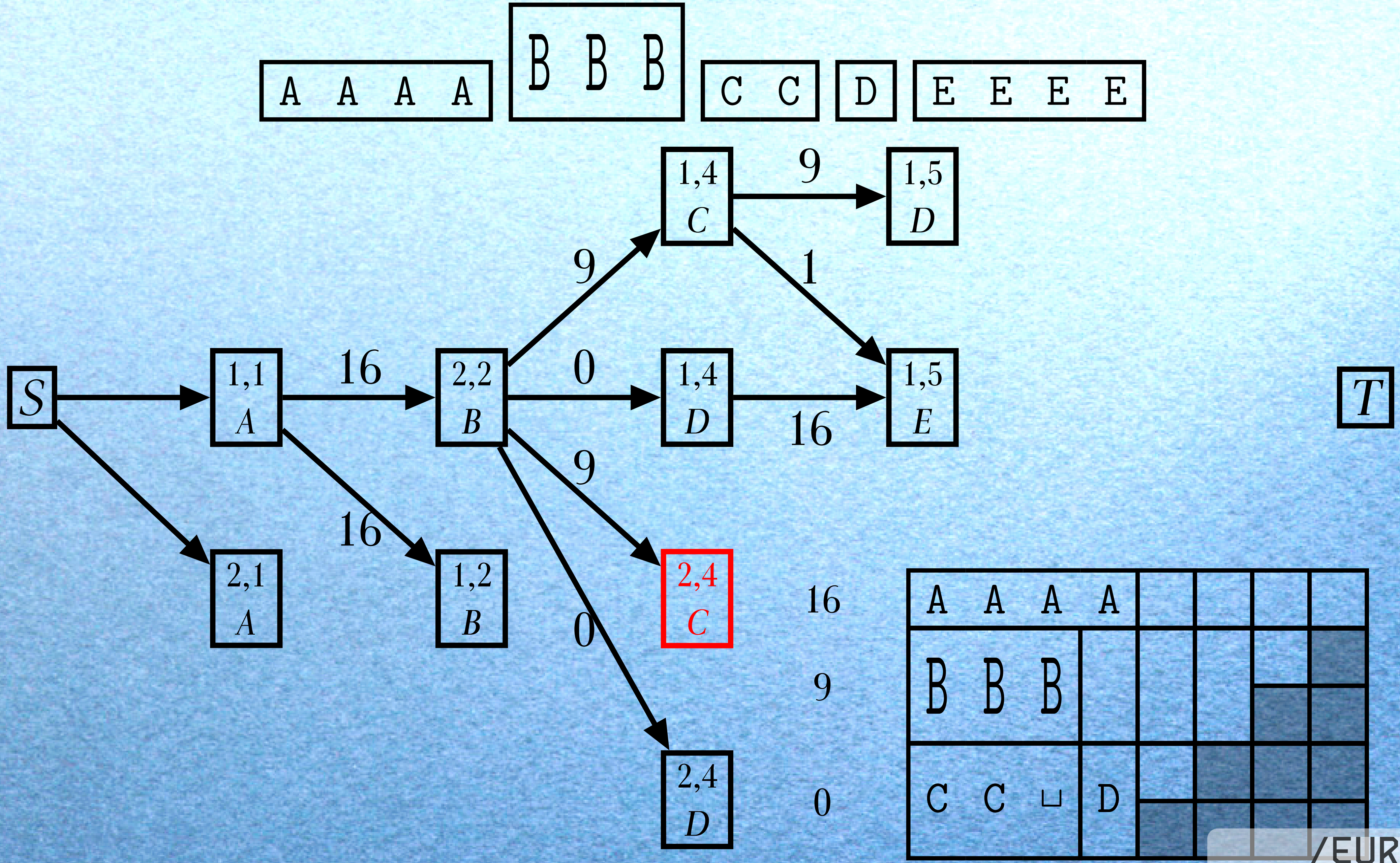
A A A A | B B B | C C | D | E E E E



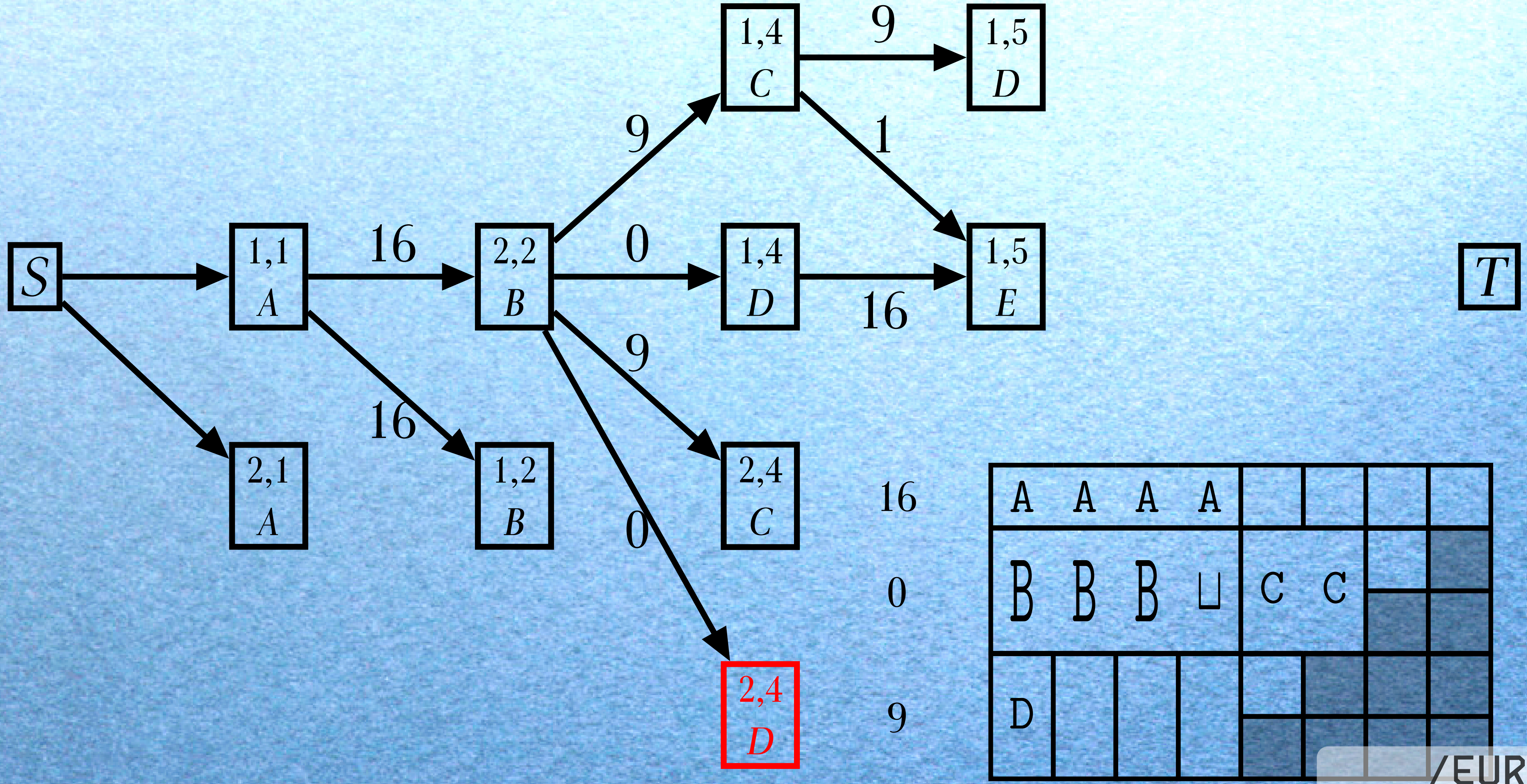
16	A	A	A	A				
0	B B B ␣				C C			
16	D							

A A A A B B B C C D E E E E

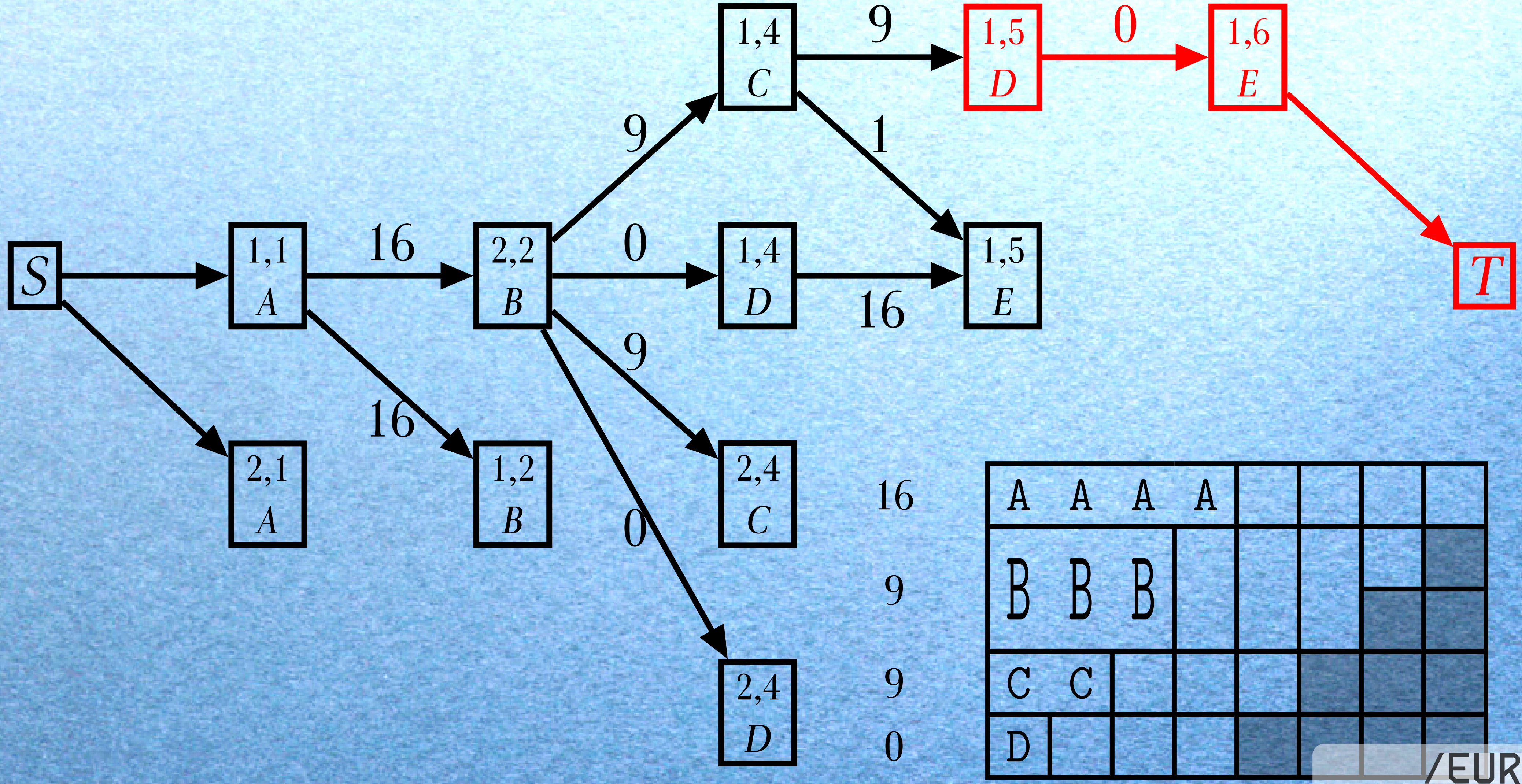


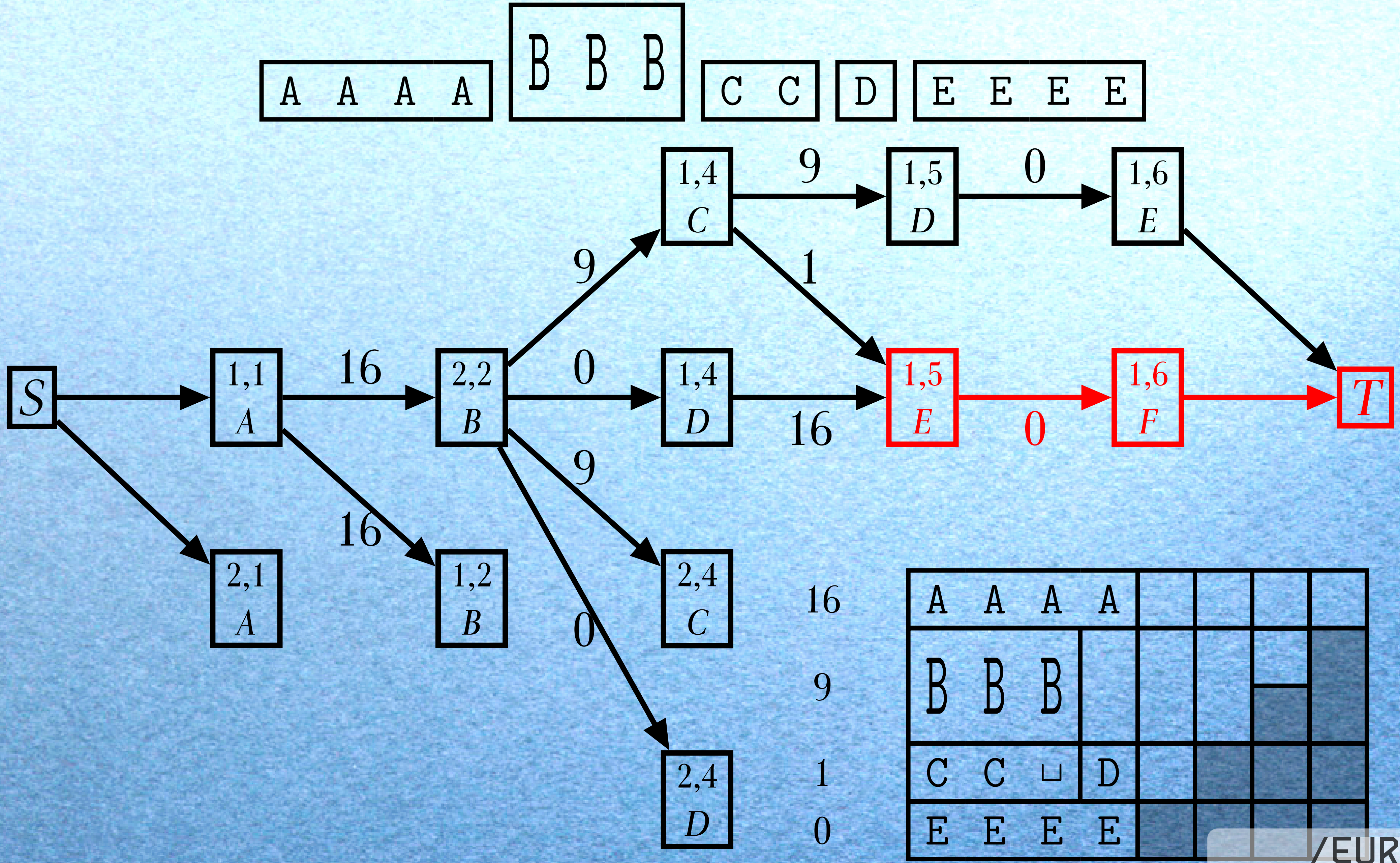

A A A A
B B B
C C
D
E E E E
1,4 C
9
1,5 D
9
1
S
1,1 A
16
2,2 B
0
1,4 D
16
1,5 E
T
9
16
2,1 A
1,2 B
0
2,4 C
2,4 D
16
9
0
A A A A
B B B
C C ␣
D

A A A A B B B C C D E E E E

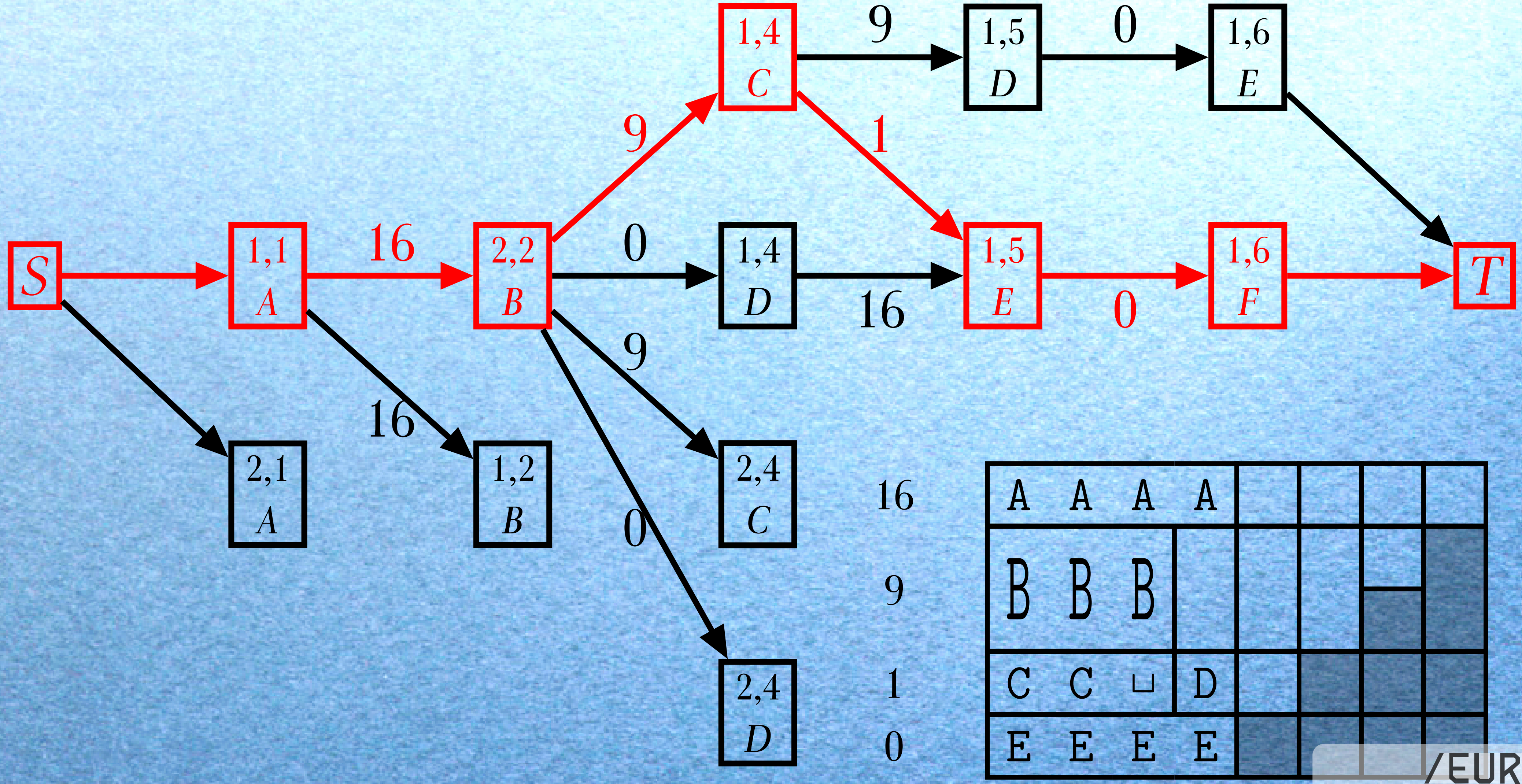


A A A A B B B C C D E E E E



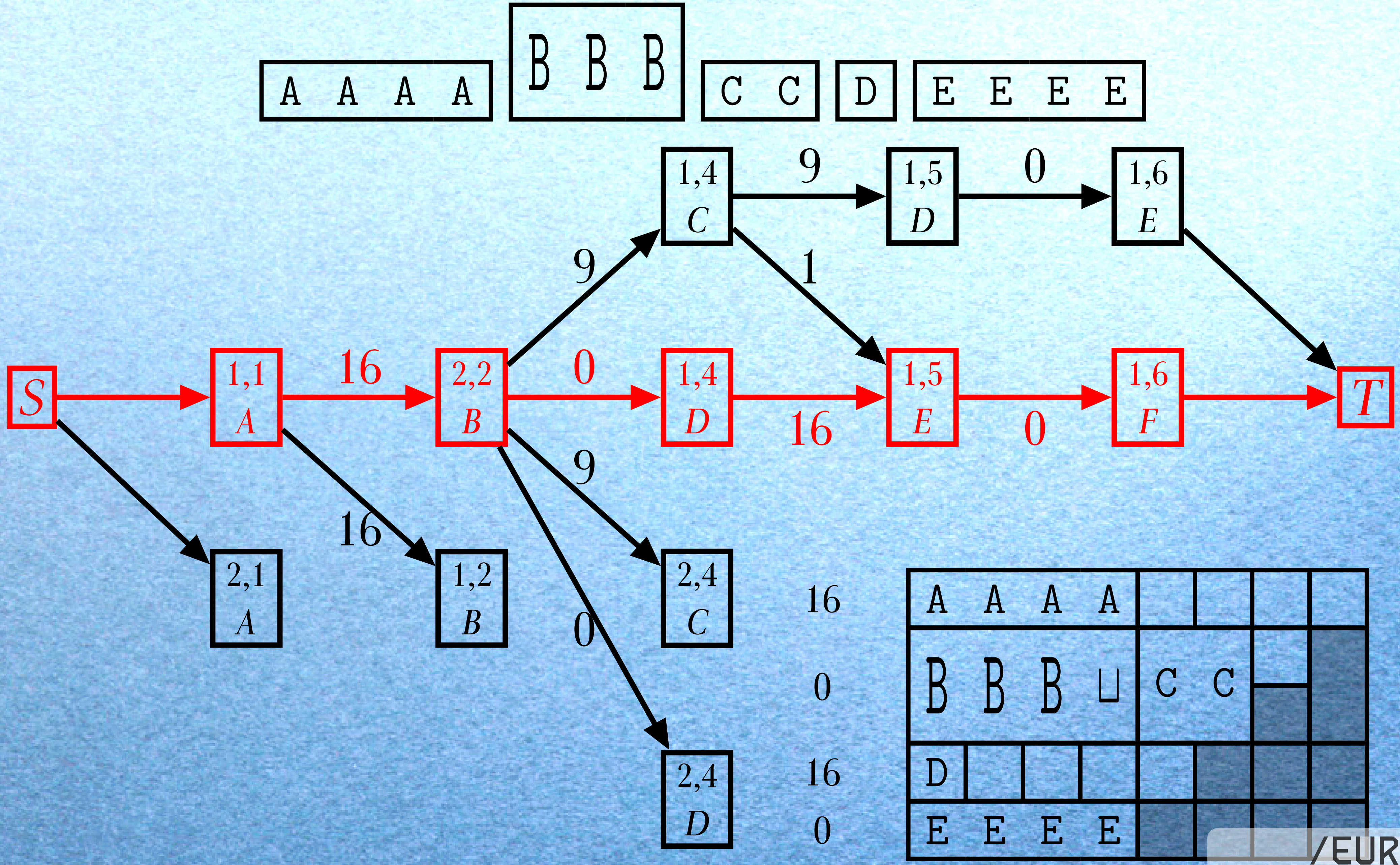

A A A A
B B B
C C
D
E E E E
S
1,1 A
2,1 A
16
16
2,2 B
1,2 B
9
0
9
0
1,4 C
1,4 D
2,4 C
2,4 D
9
1
16
1,5 D
1,5 E
0
0
1,6 E
1,6 F
T
16
9
1
0
A A A A
B B B
C C ␣ D
E E E E

A A A A | B B B | C C | D | E E E E

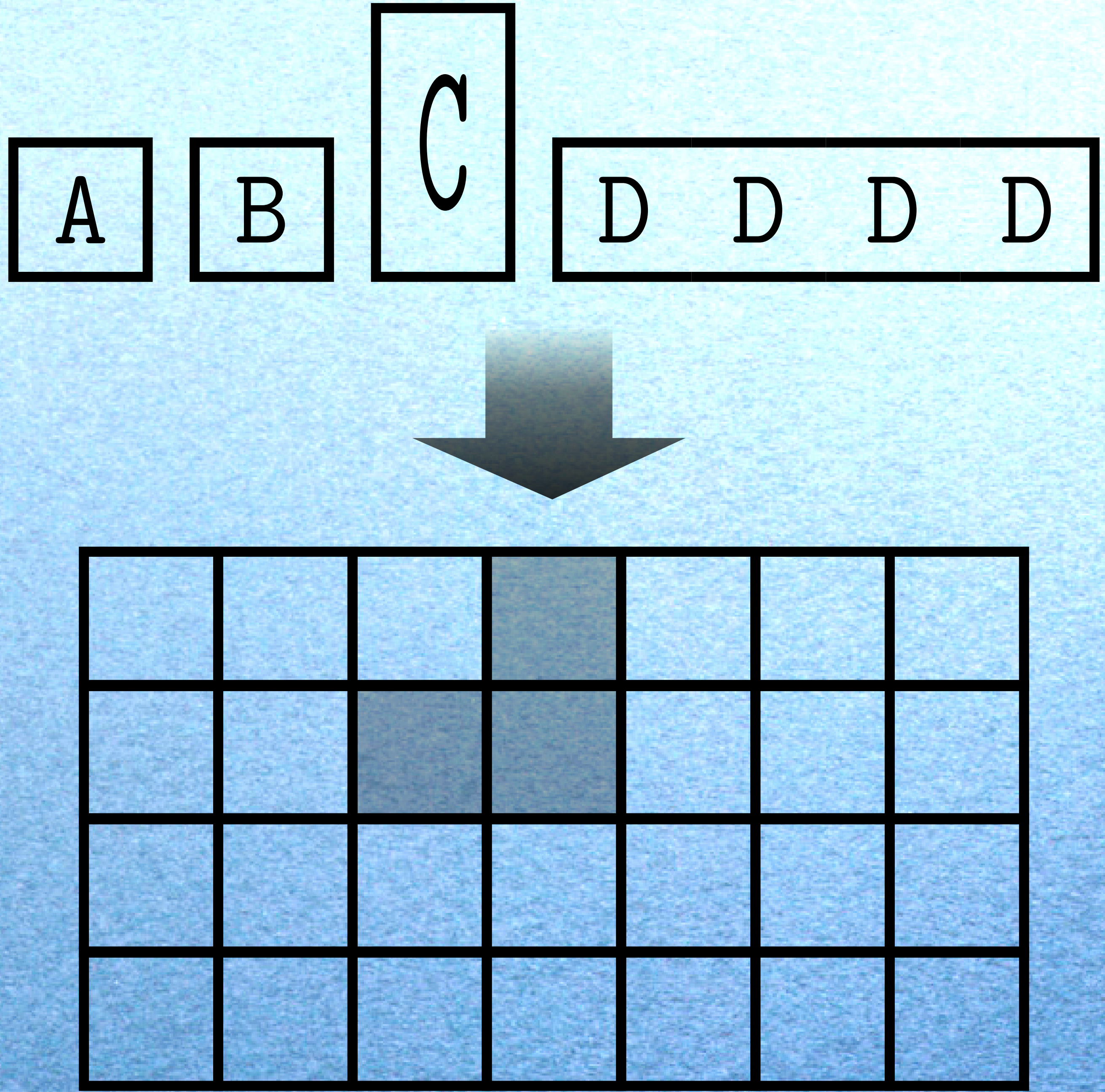


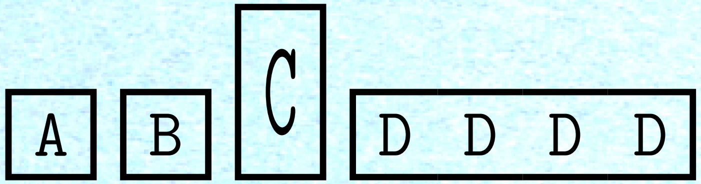
16	A A A A				
9	B B B			▨	▨
1	C C ␣ D		▨	▨	▨
0	E E E E	▨	▨	▨	▨

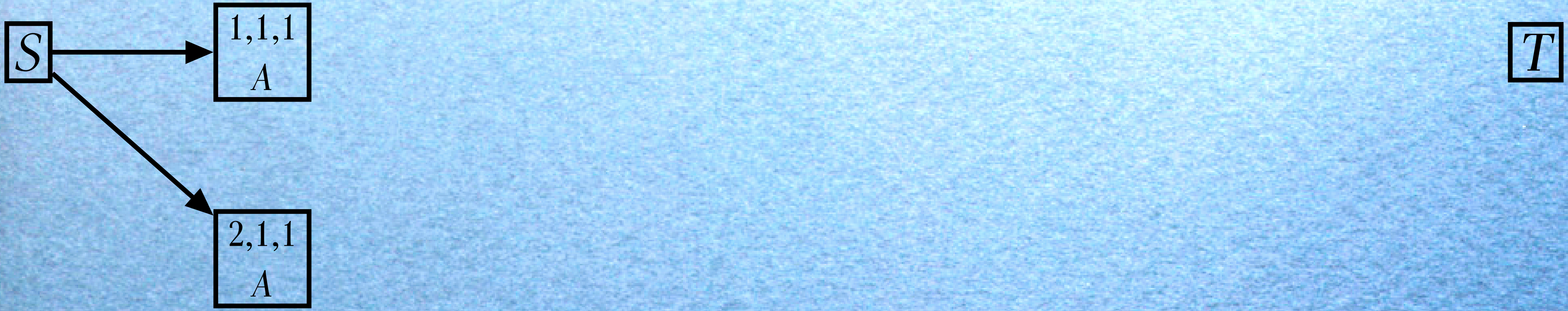
algorithm → example 4 → optimal solution

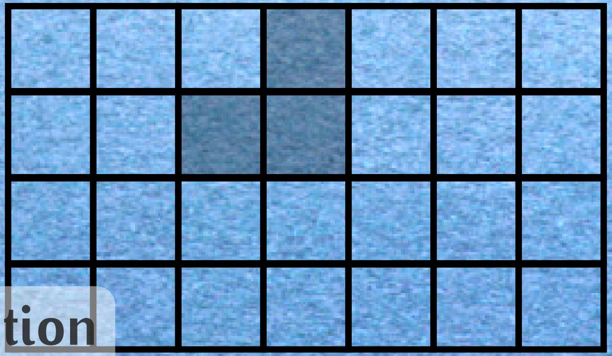


algorithm → example 4 → naïve solution

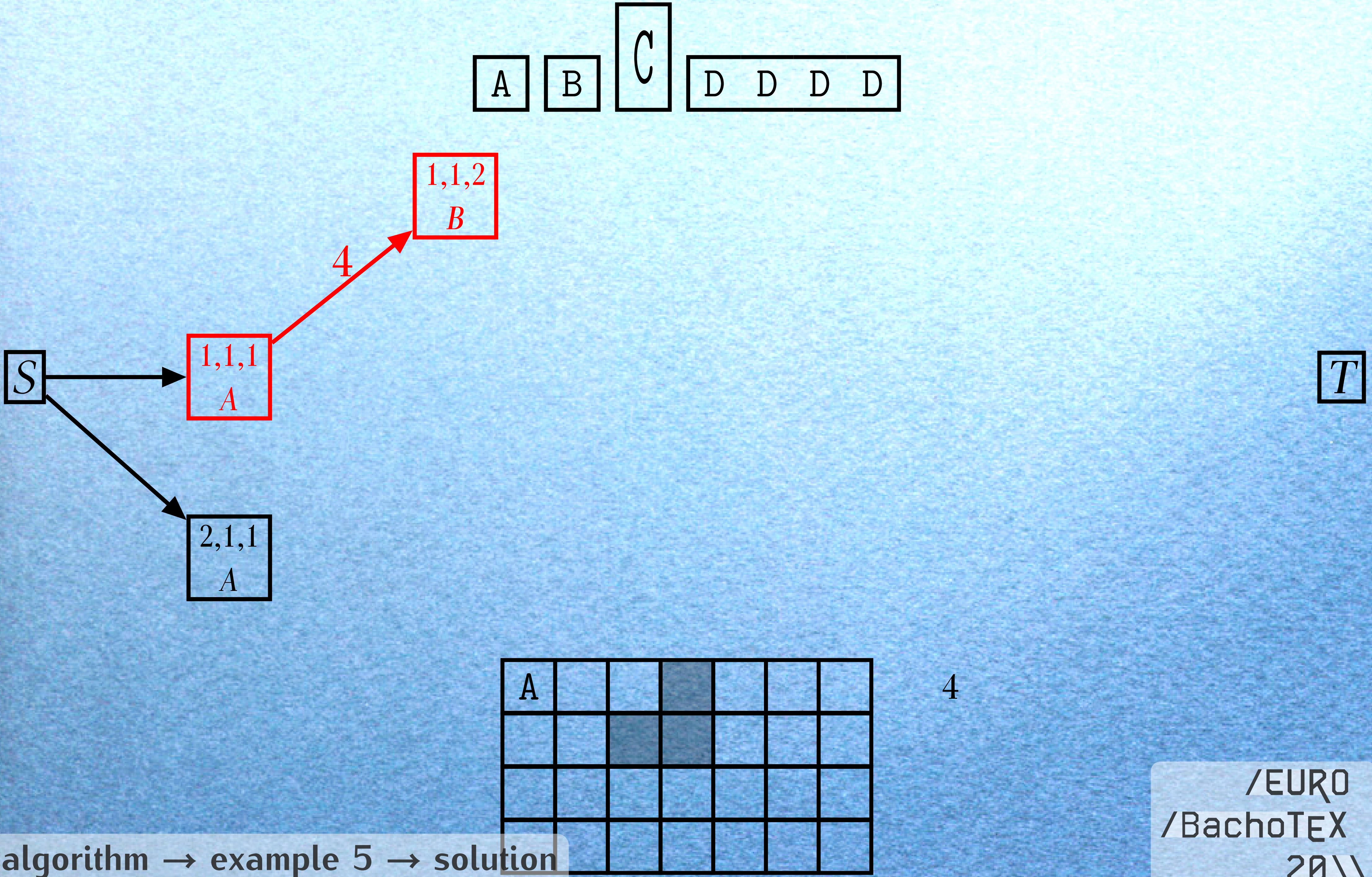

A
B
C
D
D
D
D

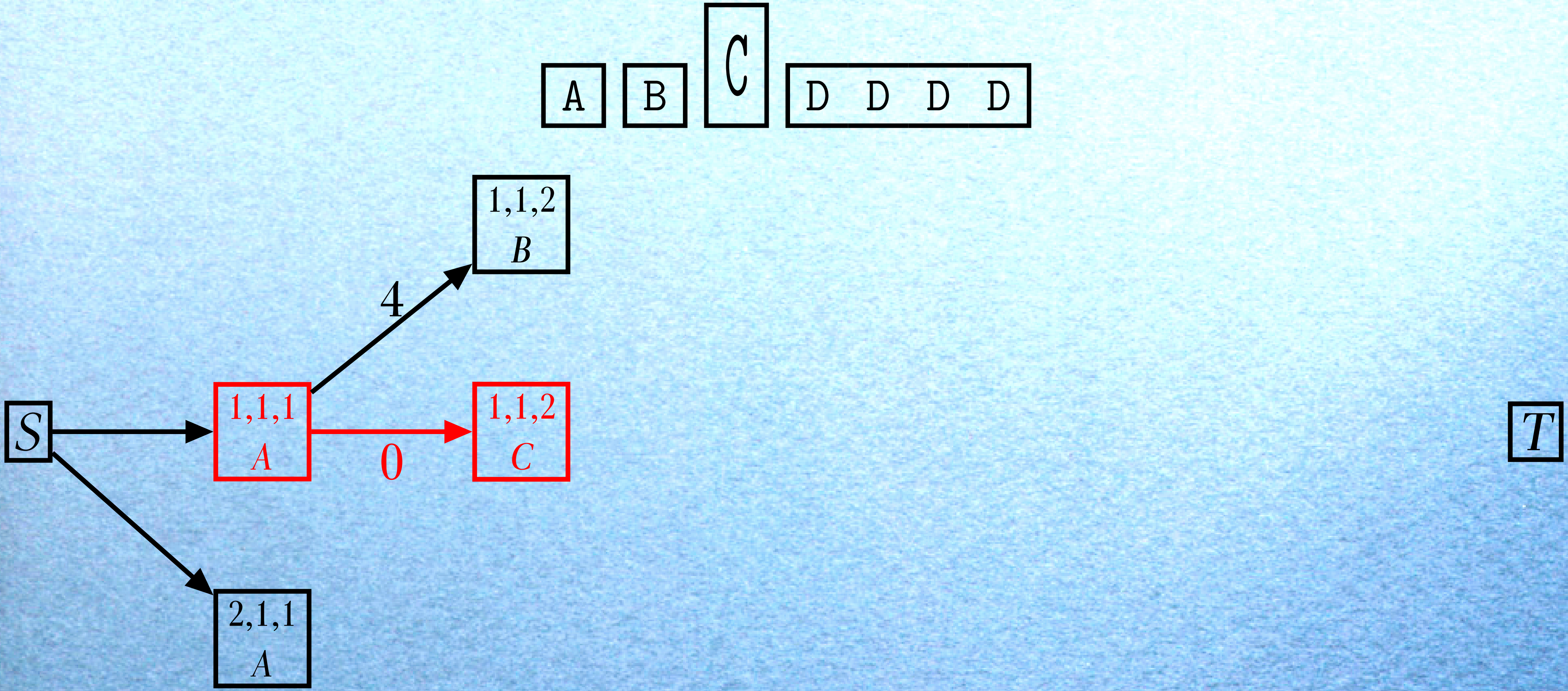


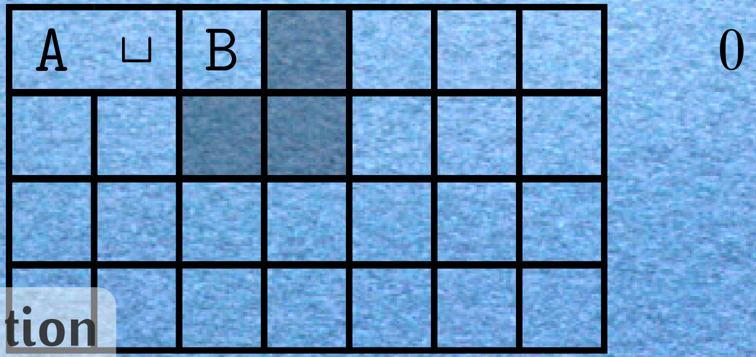




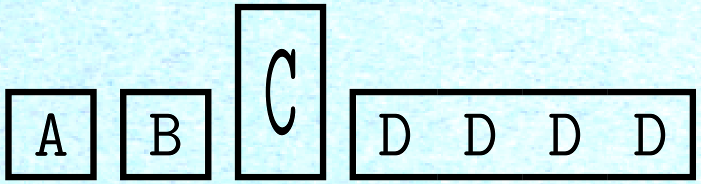
algorithm → example 5 → solution

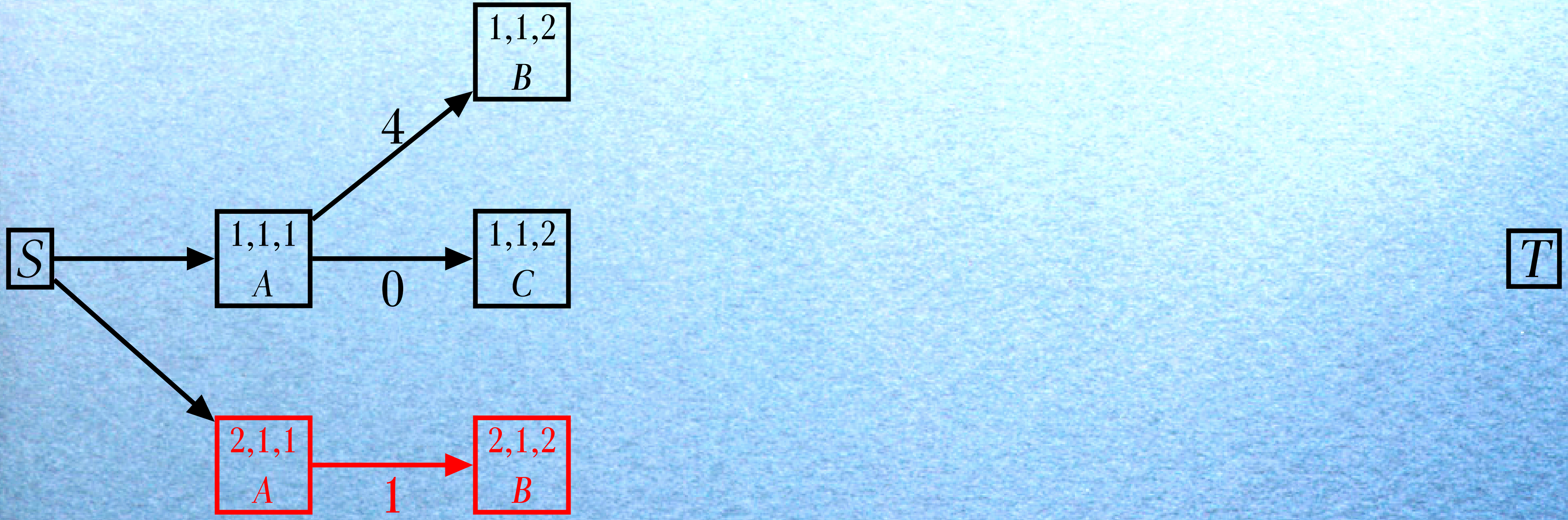

A
B
C
D D D D
1,1,2
B
4
S
1,1,1
A
T
2,1,1
A
A
4

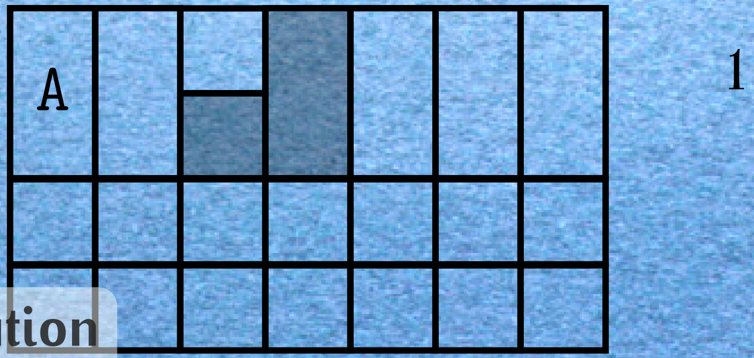




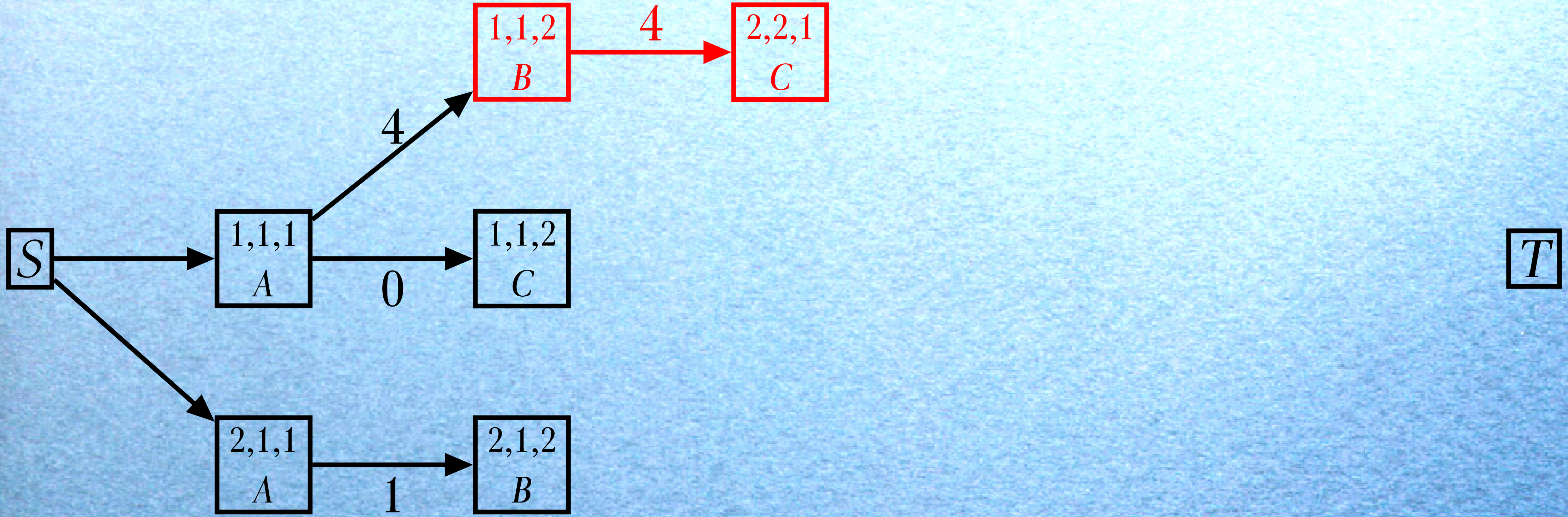
algorithm → example 5 → solution


A
B
C
D D D D


1,1,2
B
4
S
1,1,1
A
0
1,1,2
C
T
2,1,1
A
1
2,1,2
B

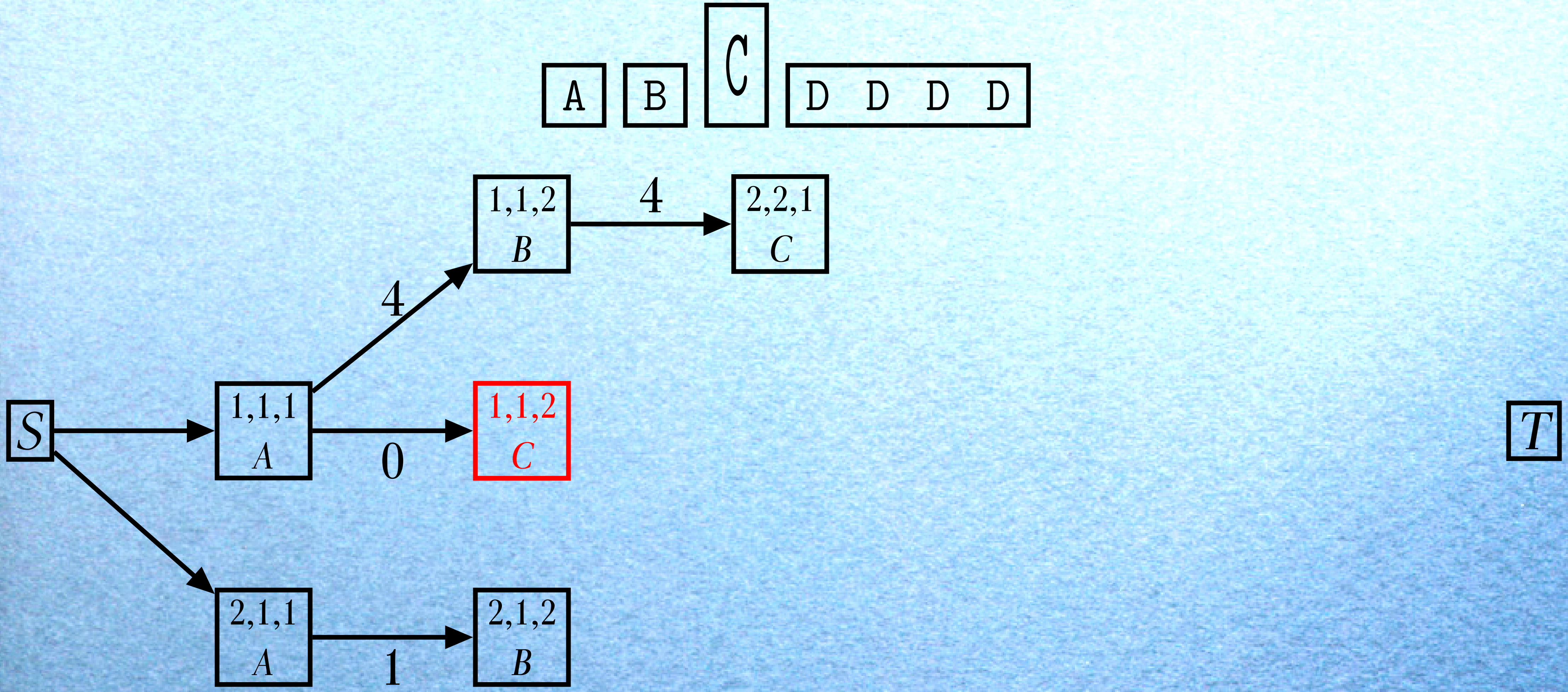

A
1

A B C D D D D



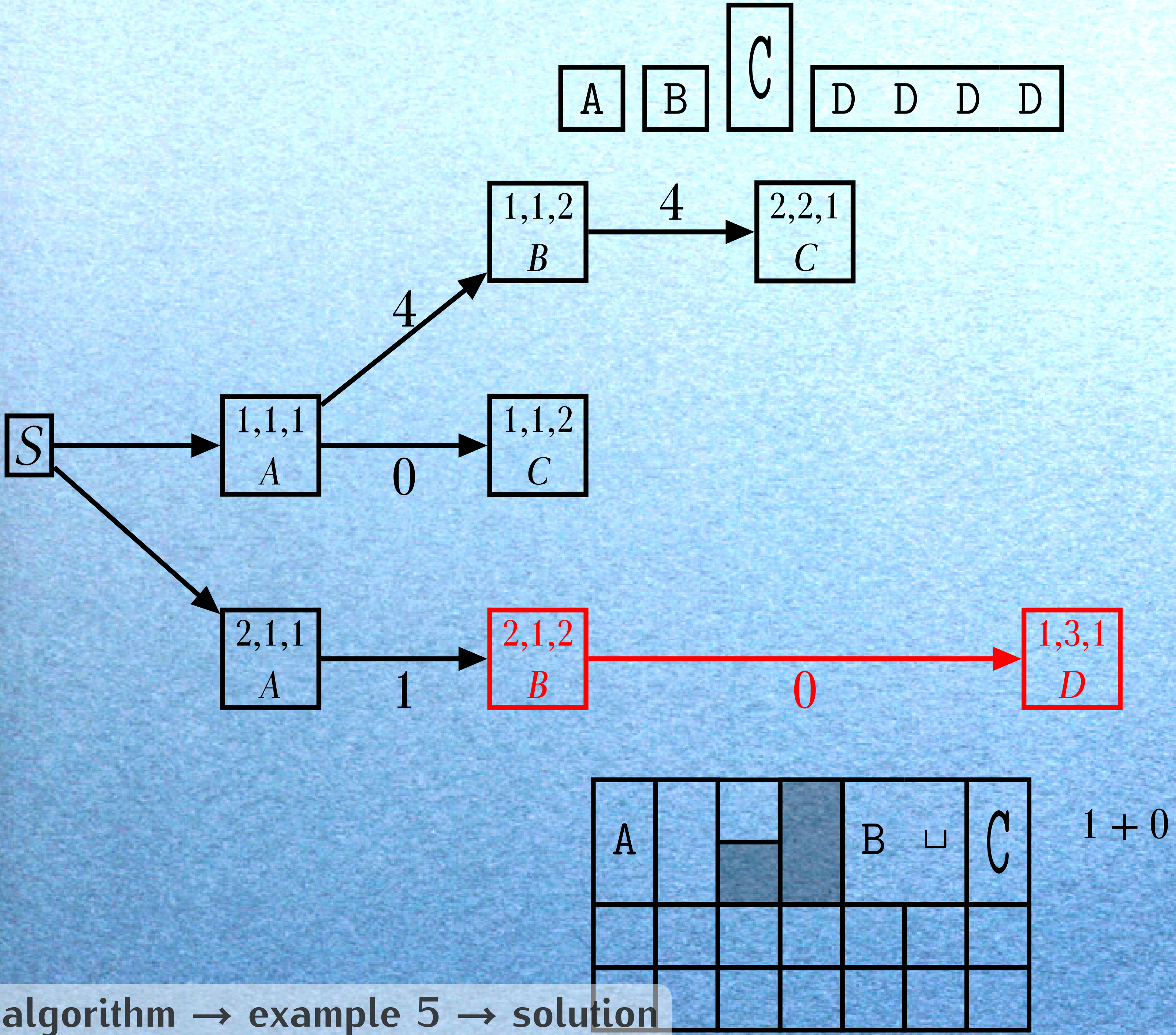
A			▨	B		
		▨	▨			

4 + 4



A	␣	B	▨			
		▨	▨			

0



A B C D D D D

S

1,1,1 *A*

2,1,1 *A*

1,1,2 *B*

1,1,2 *C*

2,2,1 *C*

2,1,2 *B*

1,3,1 *D*

T

4

4

0

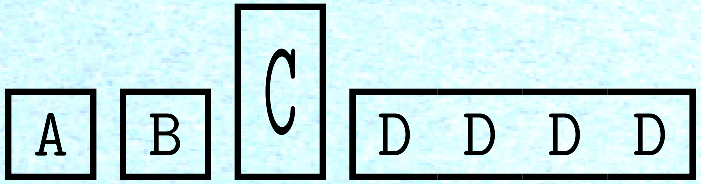
1

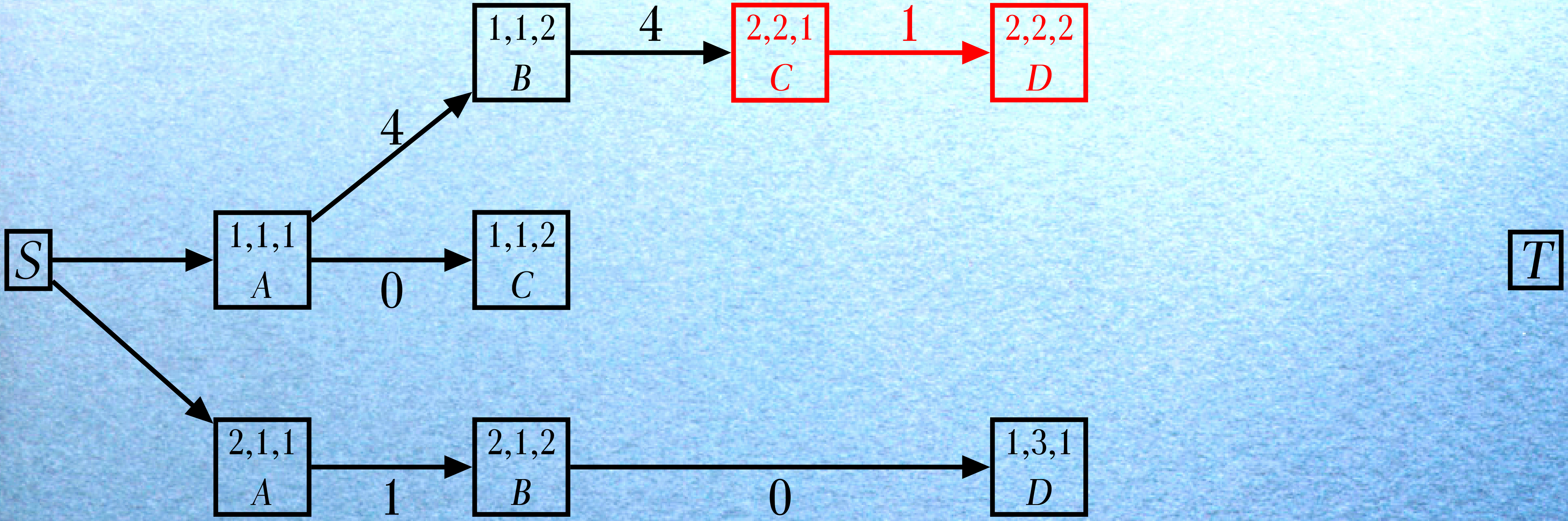
0

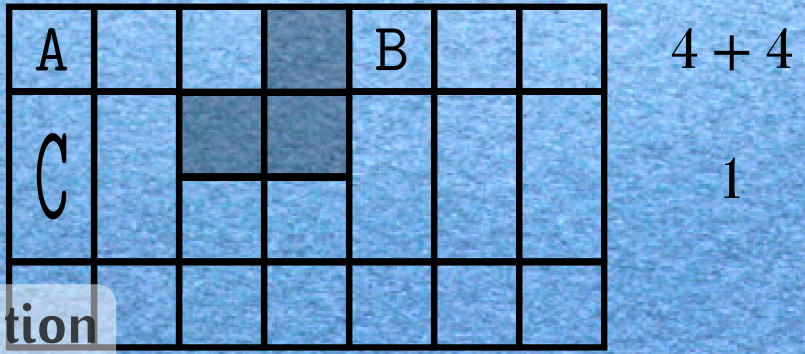
A B ␣ C

1 + 0

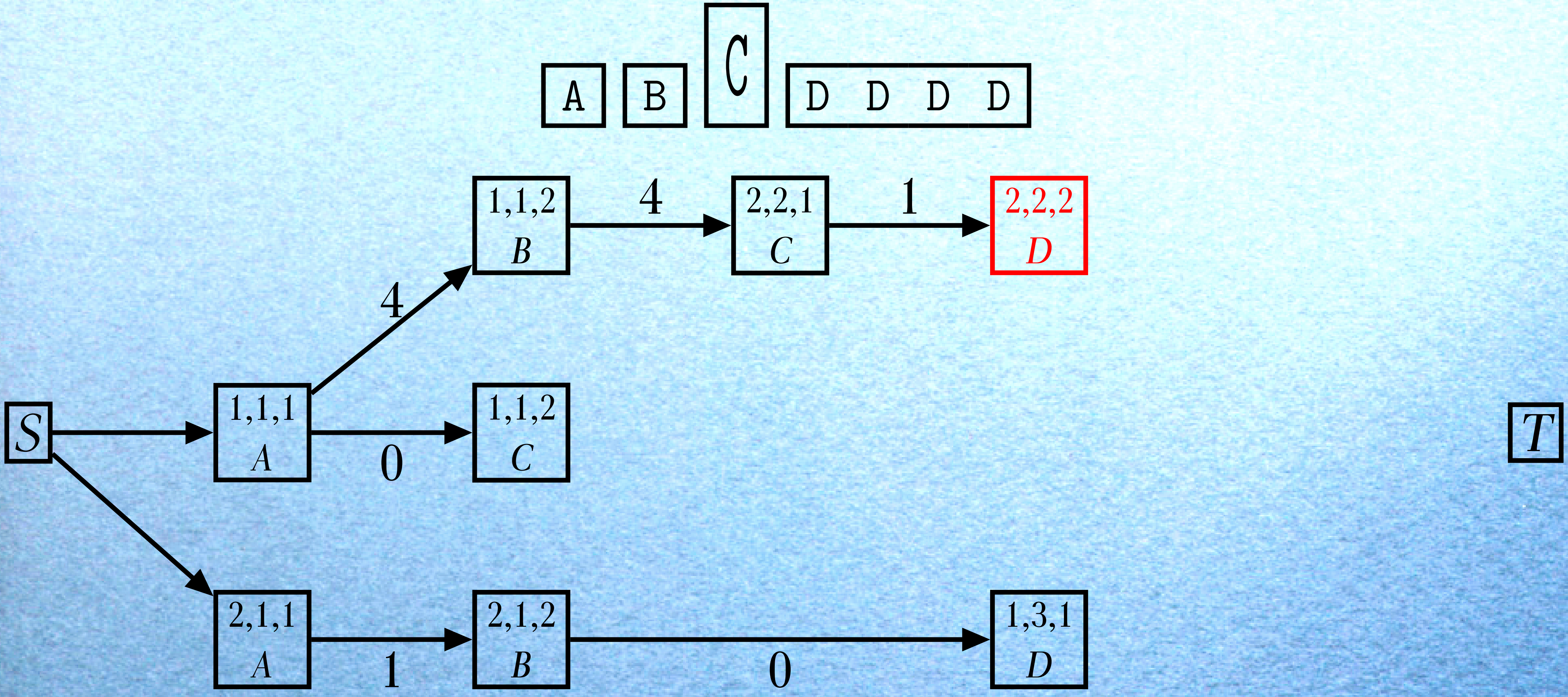
algorithm → example 5 → solution

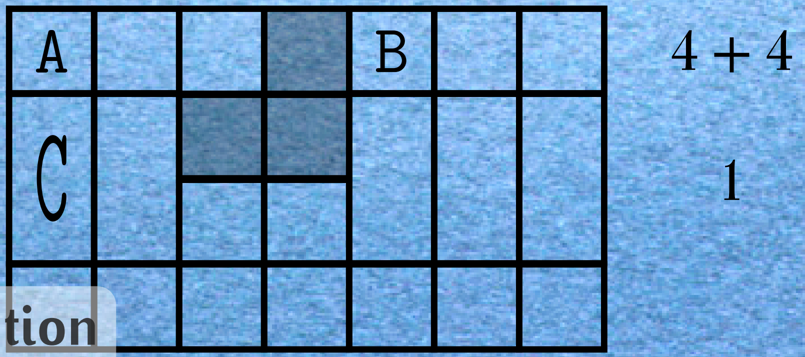




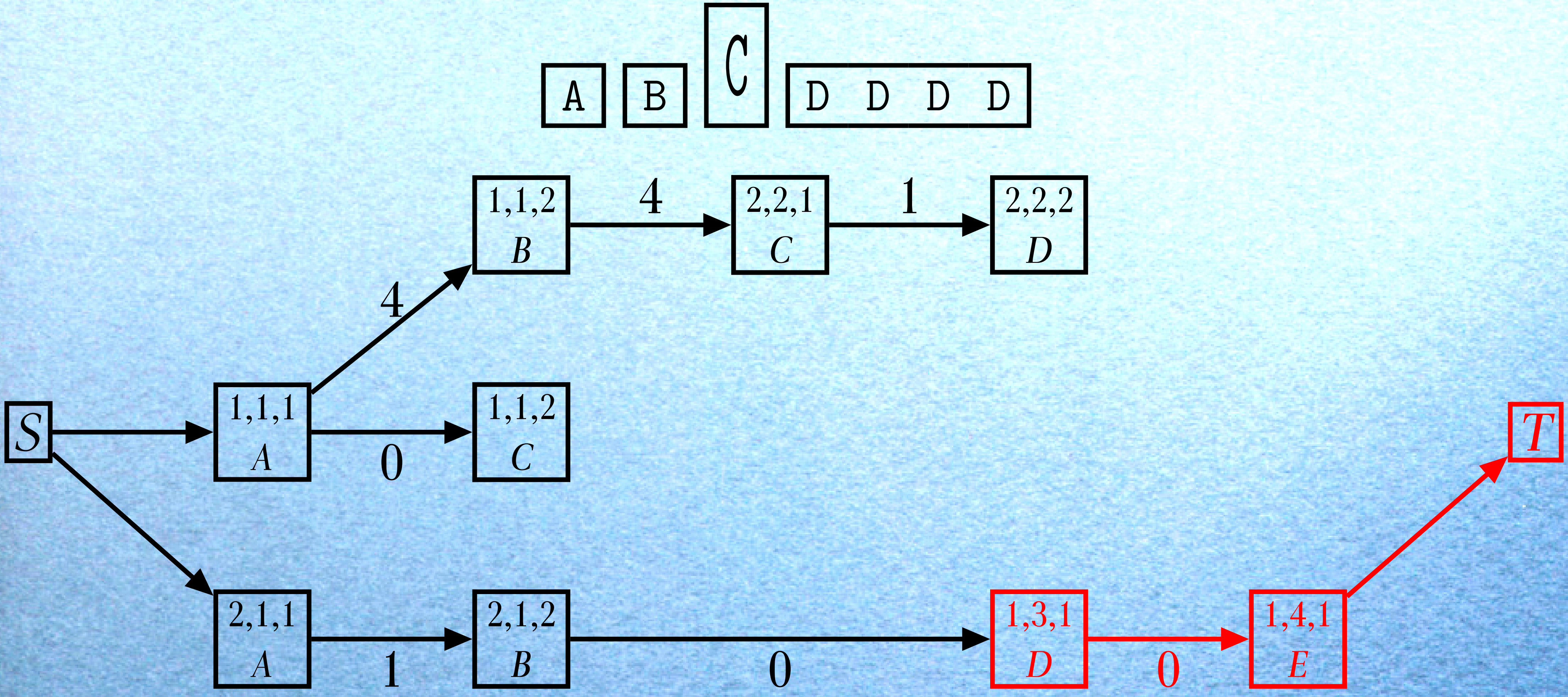


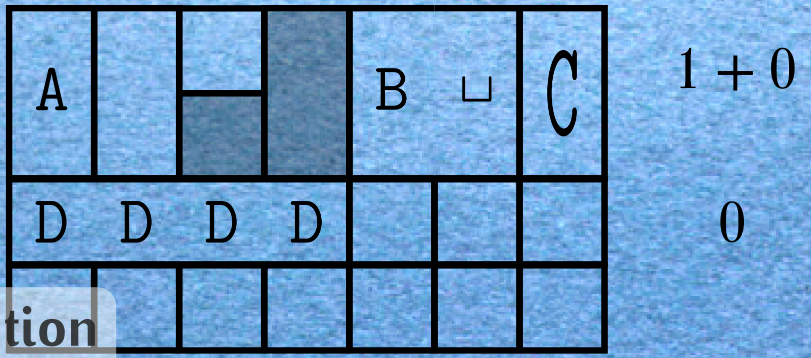
algorithm → example 5 → solution

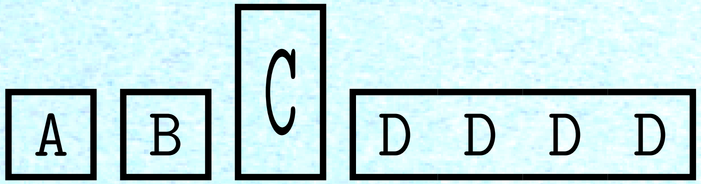


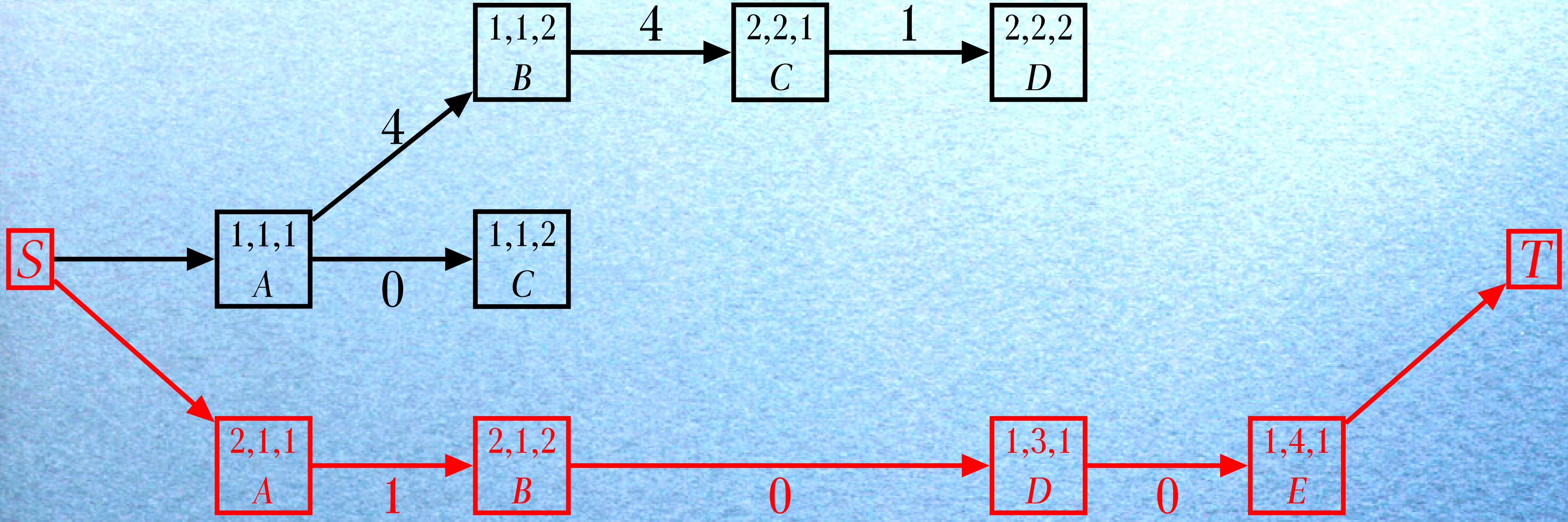


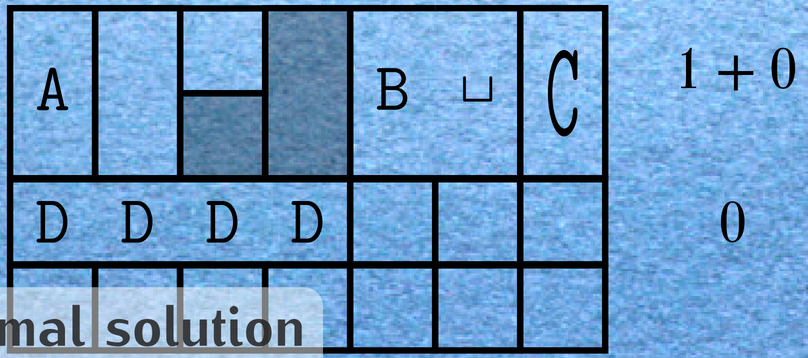
algorithm → example 5 → solution


A
B
C
D D D D
S
1,1,1 A
2,1,1 A
1,1,2 B
1,1,2 C
2,1,2 B
2,2,1 C
2,2,2 D
1,3,1 D
1,4,1 E
T
4
4
1
0
1
0
0

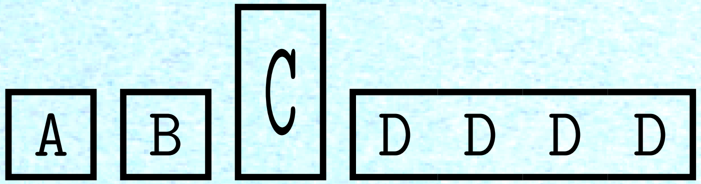

A
B ␣
C
D D D D
1 + 0
0

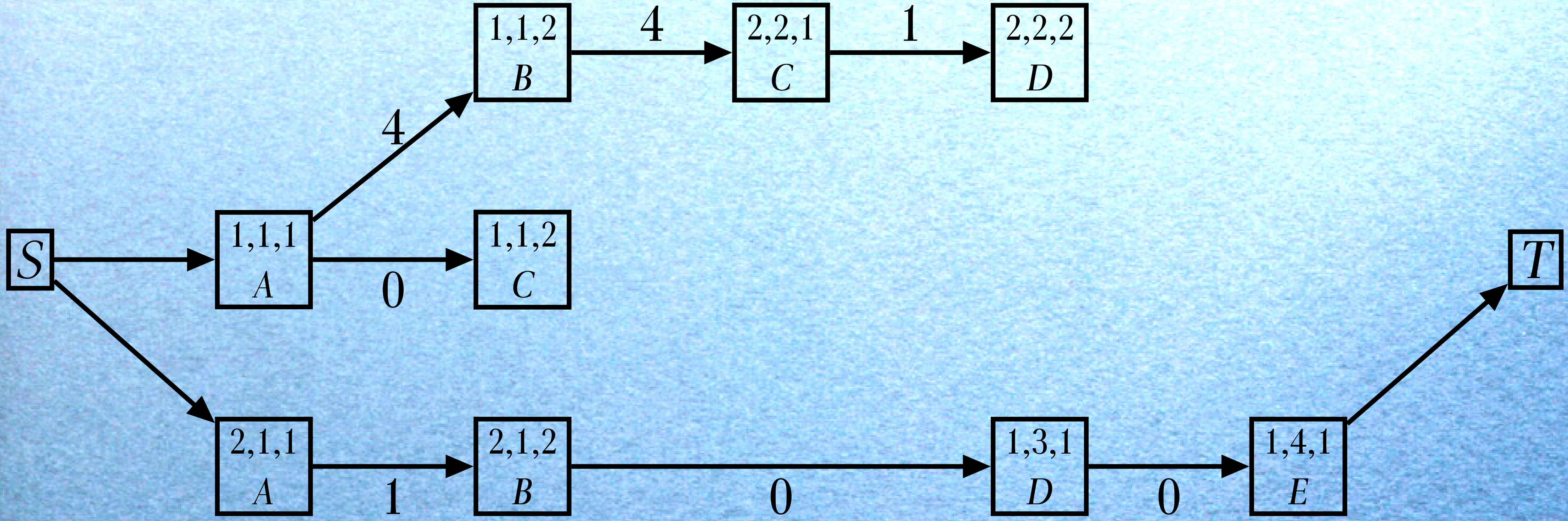


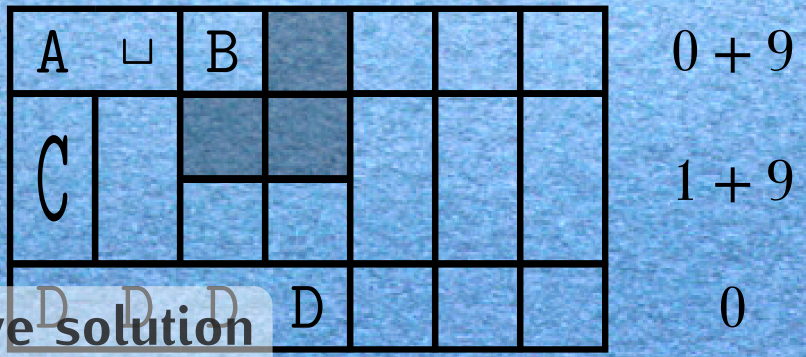




algorithm → example 5 → optimal solution







algorithm → example 5 → naïve solution

```
processNonLastSegmentNode($n$):
for $b = b_n$ to $\#B_n$ do
    for $v = 1$ to numberOfHyphenationVariantsOfBox($b$) do
        $w =$ widthOfBoxes($b_n, v_n, b, v$)
        $w_{\min} =$ minimalUsedSpaceOfSegment($\ell_n, s_n$)
        $w_{\max} =$ widthOfSegment($\ell_n, s_n$)
        if $w \geqslant w_{\min}$ and $w \leqslant w_{\max}$ then
            pushNewNodeOrUpdateExisting($Q, h_n, \ell_n, s_n + 1, b_n + 1, v_n, n$)
```

```
processNonLastSegmentNode($n$):
for $b = b_n$ to $\#B_n$ do
    for $v = 1$ to numberOfHyphenationVariantsOfBox($b$) do    $O(1)$
        $w = $ widthOfBoxes($b_n, v_n, b, v$)
        $w_{\min} = $ minimalUsedSpaceOfSegment($\ell_n, s_n$)    $O(1)$
        $w_{\max} = $ widthOfSegment($\ell_n, s_n$)    $O(1)$
        if $w \geqslant w_{\min}$ and $w \leqslant w_{\max}$ then
            pushNewNodeOrUpdateExisting($Q, h_n, \ell_n, s_n + 1, b_n + 1, v_n, n$)
```

```
processNonLastSegmentNode($n$):
for $b = b_n$ to $\#B_n$ do
    for $v = 1$ to numberOfHyphenationVariantsOfBox($b$) do    $O(1)$
        $w$ = widthOfBoxes($b_n, v_n, b, v$)    $O(1)$
        $w_{\min}$ = minimalUsedSpaceOfSegment($\ell_n, s_n$)    $O(1)$
        $w_{\max}$ = widthOfSegment($\ell_n, s_n$)    $O(1)$
        if $w \geqslant w_{\min}$ and $w \leqslant w_{\max}$ then
            pushNewNodeOrUpdateExisting($Q, h_n, \ell_n, s_n + 1, b_n + 1, v_n, n$)
```

```
processNonLastSegmentNode($n$):
for $b = b_n$ to $\#B_n$ do
    for $v = 1$ to numberOfHyphenationVariantsOfBox($b$) do    $O(1)$
        $w$ = widthOfBoxes($b_n, v_n, b, v$)    $O(1)$
        $w_{\min}$ = minimalUsedSpaceOfSegment($\ell_n, s_n$)    $O(1)$
        $w_{\max}$ = widthOfSegment($\ell_n, s_n$)    $O(1)$
        if $w \geqslant w_{\min}$ and $w \leqslant w_{\max}$ then    $O(\lg \#N_n)$
            pushNewNodeOrUpdateExisting($Q, h_n, \ell_n, s_n + 1, b_n + 1, v_n, n$)
```

algorithm → complexity analysis → ‘processNonLastSegmentNode’

```
processNonLastSegmentNode($n$):
for $b = b_n$ to $\#B_n$ do
    for $v = 1$ to numberOfHyphenationVariantsOfBox($b$) do    $O(1)$
        $w$ = widthOfBoxes($b_n, v_n, b, v$)    $O(1)$
        $w_{\min}$ = minimalUsedSpaceOfSegment($\ell_n, s_n$)    $O(1)$
        $w_{\max}$ = widthOfSegment($\ell_n, s_n$)    $O(1)$
        if $w \geqslant w_{\min}$ and $w \leqslant w_{\max}$ then    $O(\lg \#N_n)$
            pushNewNodeOrUpdateExisting($Q, h_n, \ell_n, s_n + 1, b_n + 1, v_n, n$)
```

total complexity: $O(\#B_n \cdot \lg \#N_n)$

algorithm → complexity analysis → ‘processNonLastSegmentNode’

```
processLastSegmentNode($n$):
$h_{\text{min}} = \text{computeMinimalBoxesHeight}(\ell_n, b_n, v_n)$
$h_{\text{max}} = \text{computeMaximalBoxesHeight}(\ell_n, b_n, v_n)$
for $b = b_n$ to $\#B_n$ do
    for $v = 1$ to $\text{numberOfHyphenationVariantsOfBox}(b)$ do
        if $a_n = h_n$ or $\text{heightOfBoxes}(b_n, v_n, b, v) = h_n$ then
            $w = \text{widthOfBoxes}(b_n, v_n, b, v)$
            $w_{\text{min}} = \text{minimalUsedSpaceOfSegment}(\ell_n, s_n)$
            $w_{\text{max}} = \text{widthOfSegment}(\ell_n, s_n)$
            if $w \geqslant w_{\text{min}}$ and $w \leqslant w_{\text{max}}$ then
                for $h = h_{\text{min}}$ to $h_{\text{max}}$ do
                    $\text{pushNewNodeOrUpdateExisting}(Q, h, \ell_n{+}h_n, 1, b_n{+}1, v_n, n)$
```

```
processLastSegmentNode($n$):
$h_{\min}$ = computeMinimalBoxesHeight($\ell_n, b_n, v_n$)   $O(\#B_n)$
$h_{\max}$ = computeMaximalBoxesHeight($\ell_n, b_n, v_n$)   $O(\#B_n)$
for $b = b_n$ to $\#B_n$ do
    for $v = 1$ to numberOfHyphenationVariantsOfBox($b$) do
        if $a_n = h_n$ or heightOfBoxes($b_n, v_n, b, v$) $= h_n$ then
            $w$ = widthOfBoxes($b_n, v_n, b, v$)
            $w_{\min}$ = minimalUsedSpaceOfSegment($\ell_n, s_n$)
            $w_{\max}$ = widthOfSegment($\ell_n, s_n$)
            if $w \geqslant w_{\min}$ and $w \leqslant w_{\max}$ then
                for $h = h_{\min}$ to $h_{\max}$ do
                    pushNewNodeOrUpdateExisting($Q, h, \ell_n{+}h_n, 1, b_n{+}1, v_n, n$)
```

```
processLastSegmentNode($n$):
$h_{\min} = \text{computeMinimalBoxesHeight}(\ell_n, b_n, v_n)$    $O(\#B_n)$
$h_{\max} = \text{computeMaximalBoxesHeight}(\ell_n, b_n, v_n)$    $O(\#B_n)$
for $b = b_n$ to $\#B_n$ do
    for $v = 1$ to numberOfHyphenationVariantsOfBox($b$) do    $O(1)$
        if $a_n = h_n$ or $\text{heightOfBoxes}(b_n, v_n, b, v) = h_n$ then
            $w = \text{widthOfBoxes}(b_n, v_n, b, v)$
            $w_{\min} = \text{minimalUsedSpaceOfSegment}(\ell_n, s_n)$    $O(1)$
            $w_{\max} = \text{widthOfSegment}(\ell_n, s_n)$    $O(1)$
            if $w \geqslant w_{\min}$ and $w \leqslant w_{\max}$ then
                for $h = h_{\min}$ to $h_{\max}$ do
                    pushNewNodeOrUpdateExisting($Q, h, \ell_n{+}h_n, 1, b_n{+}1, v_n, n$)
```

```
processLastSegmentNode($n$):
$h_{\min}$ = computeMinimalBoxesHeight($\ell_n, b_n, v_n$)   $O(\#B_n)$
$h_{\max}$ = computeMaximalBoxesHeight($\ell_n, b_n, v_n$)   $O(\#B_n)$
for $b = b_n$ to $\#B_n$ do
    for $v = 1$ to numberOfHyphenationVariantsOfBox($b$) do   $O(1)$
        if $a_n = h_n$ or heightOfBoxes($b_n, v_n, b, v$) $= h_n$ then   $O(1)$
            $w$ = widthOfBoxes($b_n, v_n, b, v$)   $O(1)$
            $w_{\min}$ = minimalUsedSpaceOfSegment($\ell_n, s_n$)   $O(1)$
            $w_{\max}$ = widthOfSegment($\ell_n, s_n$)   $O(1)$
            if $w \geqslant w_{\min}$ and $w \leqslant w_{\max}$ then
                for $h = h_{\min}$ to $h_{\max}$ do
                    pushNewNodeOrUpdateExisting($Q, h, \ell_n+h_n, 1, b_n+1, v_n, n$)
```

```
processLastSegmentNode($n$):
$h_{\min} = \mathrm{computeMinimalBoxesHeight}(\ell_n, b_n, v_n)$    $O(\#B_n)$
$h_{\max} = \mathrm{computeMaximalBoxesHeight}(\ell_n, b_n, v_n)$    $O(\#B_n)$
for  $b = b_n$  to  $\#B_n$  do
    for  $v = 1$  to  $\mathrm{numberOfHyphenationVariantsOfBox}(b)$  do    $O(1)$
        if  $a_n = h_n$  or  $\mathrm{heightOfBoxes}(b_n, v_n, b, v) = h_n$  then    $O(1)$
            $w = \mathrm{widthOfBoxes}(b_n, v_n, b, v)$    $O(1)$
            $w_{\min} = \mathrm{minimalUsedSpaceOfSegment}(\ell_n, s_n)$    $O(1)$
            $w_{\max} = \mathrm{widthOfSegment}(\ell_n, s_n)$    $O(1)$
            if  $w \geqslant w_{\min}$  and  $w \leqslant w_{\max}$  then
                for  $h = h_{\min}$  to  $h_{\max}$  do    $O(\lg \#N_n)$
                    $\mathrm{pushNewNodeOrUpdateExisting}(Q, h, \ell_n{+}h_n, 1, b_n{+}1, v_n, n)$
```

```
processLastSegmentNode($n$):
$h_{\min} = \text{computeMinimalBoxesHeight}(\ell_n, b_n, v_n)$   $O(\#B_n)$
$h_{\max} = \text{computeMaximalBoxesHeight}(\ell_n, b_n, v_n)$   $O(\#B_n)$
for $b = b_n$ to $\#B_n$ do
    for $v = 1$ to $\text{numberOfHyphenationVariantsOfBox}(b)$ do   $O(1)$
        if $a_n = h_n$ or $\text{heightOfBoxes}(b_n, v_n, b, v) = h_n$ then   $O(1)$
            $w = \text{widthOfBoxes}(b_n, v_n, b, v)$   $O(1)$
            $w_{\min} = \text{minimalUsedSpaceOfSegment}(\ell_n, s_n)$   $O(1)$
            $w_{\max} = \text{widthOfSegment}(\ell_n, s_n)$   $O(1)$
            if $w \geqslant w_{\min}$ and $w \leqslant w_{\max}$ then
                for $h = h_{\min}$ to $h_{\max}$ do   $O(\lg \#N_n)$
                    $\text{pushNewNodeOrUpdateExisting}(Q, h, \ell_n{+}h_n, 1, b_n{+}1, v_n, n)$
```

total complexity: $O(\#B_n \cdot \#H_n \cdot \lg \#N_n)$

```
processLastSegmentNode($S$)   // initialization
while   queueIsNotEmpty($Q$)   do
    $n$ = popNode($Q$)
    if   $h_n + \ell_n > \#L$   or   $b_n > \#B$   then
        if   $c_T > c_n$   then
            $c_T = c_n,\quad \pi_T = n$
    else
        if   isLastSegmentOfLine($n$)   then
            processLastSegmentNode($n$)
        else
            processNonLastSegmentNode($n$)
```

```
processLastSegmentNode($S$)   // initialization   $O(\#B_S \cdot \#H_S \cdot \lg \#N_S)$
while queueIsNotEmpty($Q$) do
    $n$ = popNode($Q$)
    if $h_n + \ell_n > \#L$ or $b_n > \#B$ then
        if $c_T > c_n$ then
            $c_T = c_n, \quad \pi_T = n$
    else
        if isLastSegmentOfLine($n$) then
            processLastSegmentNode($n$)   $O(\#B_n \cdot \#H_n \cdot \lg \#N_n)$
        else
            processNonLastSegmentNode($n$)   $O(\#B_n \cdot \lg \#N_n)$
```

```
processLastSegmentNode($S$)   // initialization   $O(\#B_S \cdot \#H_S \cdot \lg \#N_S)$
while  queueIsNotEmpty($Q$)  do   $O(1)$
    $n$ = popNode($Q$)
    if  $h_n + \ell_n > \#L$  or  $b_n > \#B$  then
        if  $c_T > c_n$  then
            $c_T = c_n, \quad \pi_T = n$
    else
        if  isLastSegmentOfLine($n$)  then   $O(1)$
            processLastSegmentNode($n$)   $O(\#B_n \cdot \#H_n \cdot \lg \#N_n)$
        else
            processNonLastSegmentNode($n$)   $O(\#B_n \cdot \lg \#N_n)$
```

algorithm → complexity analysis

```
processLastSegmentNode($S$)   // initialization   $O(\#B_S \cdot \#H_S \cdot \lg \#N_S)$
while  queueIsNotEmpty($Q$)  do   $O(1)$
    $n$ = popNode($Q$)   $O(\#S)$
    if  $h_n + \ell_n > \#L$  or  $b_n > \#B$  then
        if  $c_T > c_n$  then
            $c_T = c_n$,  $\pi_T = n$
    else
        if  isLastSegmentOfLine($n$)  then   $O(1)$
            processLastSegmentNode($n$)   $O(\#B_n \cdot \#H_n \cdot \lg \#N_n)$
        else
            processNonLastSegmentNode($n$)   $O(\#B_n \cdot \lg \#N_n)$
```

```
processLastSegmentNode($S$)   // initialization   $O(\#B_S \cdot \#H_S \cdot \lg \#N_S)$
while   queueIsNotEmpty($Q$)   do   $O(1)$
    $n$ = popNode($Q$)   $O(\#S)$
    if   $h_n + \ell_n > \#L$   or   $b_n > \#B$   then
        if   $c_T > c_n$   then
            $c_T = c_n$,   $\pi_T = n$
    else
        if   isLastSegmentOfLine($n$)   then   $O(1)$
            processLastSegmentNode($n$)   $O(\#B_n \cdot \#H_n \cdot \lg \#N_n)$
        else
            processNonLastSegmentNode($n$)   $O(\#B_n \cdot \lg \#N_n)$
```

total complexity: $O(\#B^2 \cdot \#H^2 \cdot \#S \cdot \lg \#B \cdot \#H \cdot \#S)$

\bye

“Thank you for your attention!”

/EURO
/BachoTEX
20\\