Dijkstra:

begin

1. $S \leftarrow \{s\}$.

2. $D[s] \leftarrow 0$.

3. for each v in $V - \{s\}$ do $D[v] \leftarrow w(s,v)$

while $S \neq V$

4. begin

5. choose a vertex $z \in V - S$ s.t. $D[z]$ is min

6. add z to S.

7. for each $v \in V - S$ do $D[v] \leftarrow \mathrm{Min}\{D[v], D[z] + w(z,v)\}$.

end

end

<u>Thm</u> Dijkstra's alg. gives the shortest path from s in time $O(n^2)$

<u>pf</u> <u>Correctness</u>: We shall prove by induction on $|S|$ that

(i) for each $v \in S$, $D[v]$ is the length of the shortest path from s. that passes through S $D[v]$ is the length of the

(ii) for each $v \notin S$, the shortest path to v passes through S except for v.

Base case is covered by lines 1-3

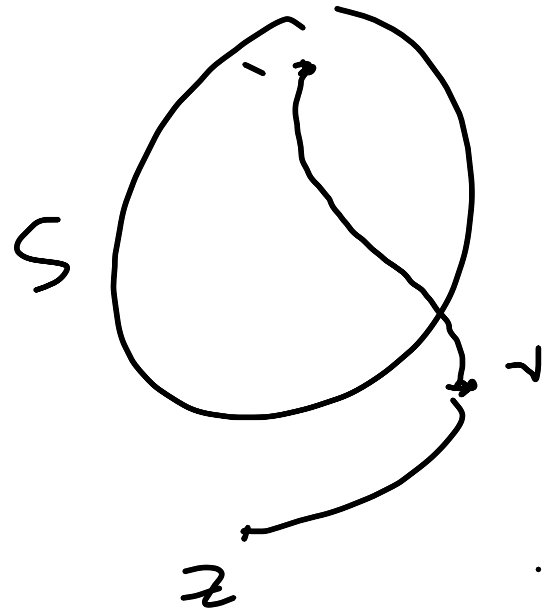
Assume that z has been added in line 5.

Suppose D[z] is not the length of the shortest path from s to z.

Let P be a shortest path from s to z

Hence P must pass through a vertext outside S different from z

Let v be the ~~fir~~ first such vertices.



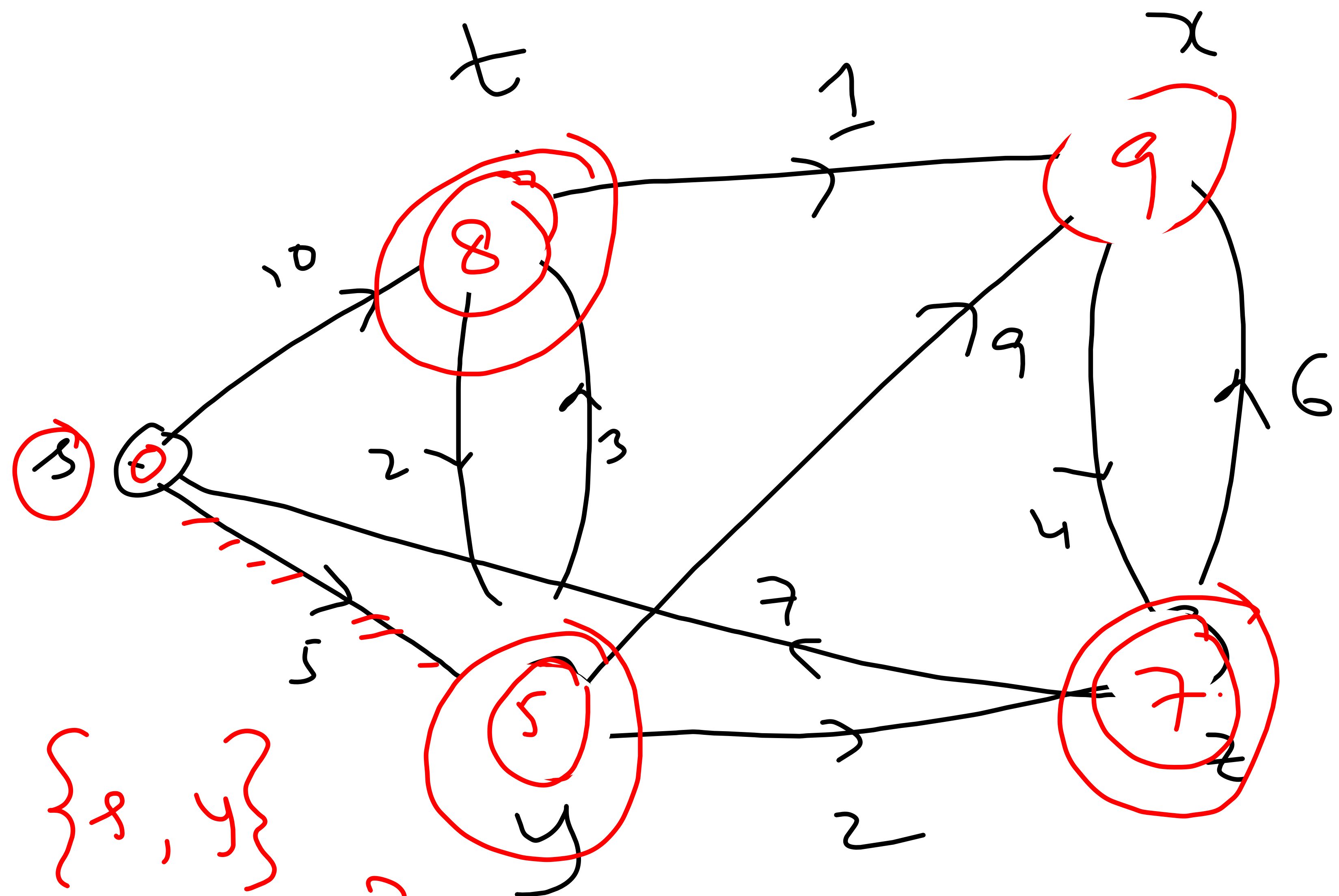
By induction hypothesis $D[v]$ is the length of the shortest path from s to v.

$\therefore D[v] \le \text{length } T < D[z]$

by our assumption

This contradicts our choice of z. Hence no such path exists

2^{nd} part of induction holds due to line 8.



$S = \{s, y\}$

$S = \{s, y, z\}$

$S = \{s, y, z, t\}$

Dynamic Programming

1. Longest Common Subsequence

Given a sequence $X = (x_1, \ldots, x_n)$

$Z = (z_1, z_2, \ldots, z_k)$ is a subsequence of X

if $\exists\ 1 \le i_1 < i_2 < \ldots < i_k \le n$ s.t.

$\times\ \times\ \times\ 0 - \cdot\ 0\ \cdot\ \partial\ \cdot \times$

$x_{i_1} = z_1$

$x_{i_2} = z_2 \quad \ldots \quad x_{i_k} = z_k.$

LCS - problem

Given 2 seq^{n}s $X = (x_1, \ldots, x_m)$
$Y = (y_1, \ldots, y_n)$, find a seq^n $Z = (z_1, \ldots, z_k)$

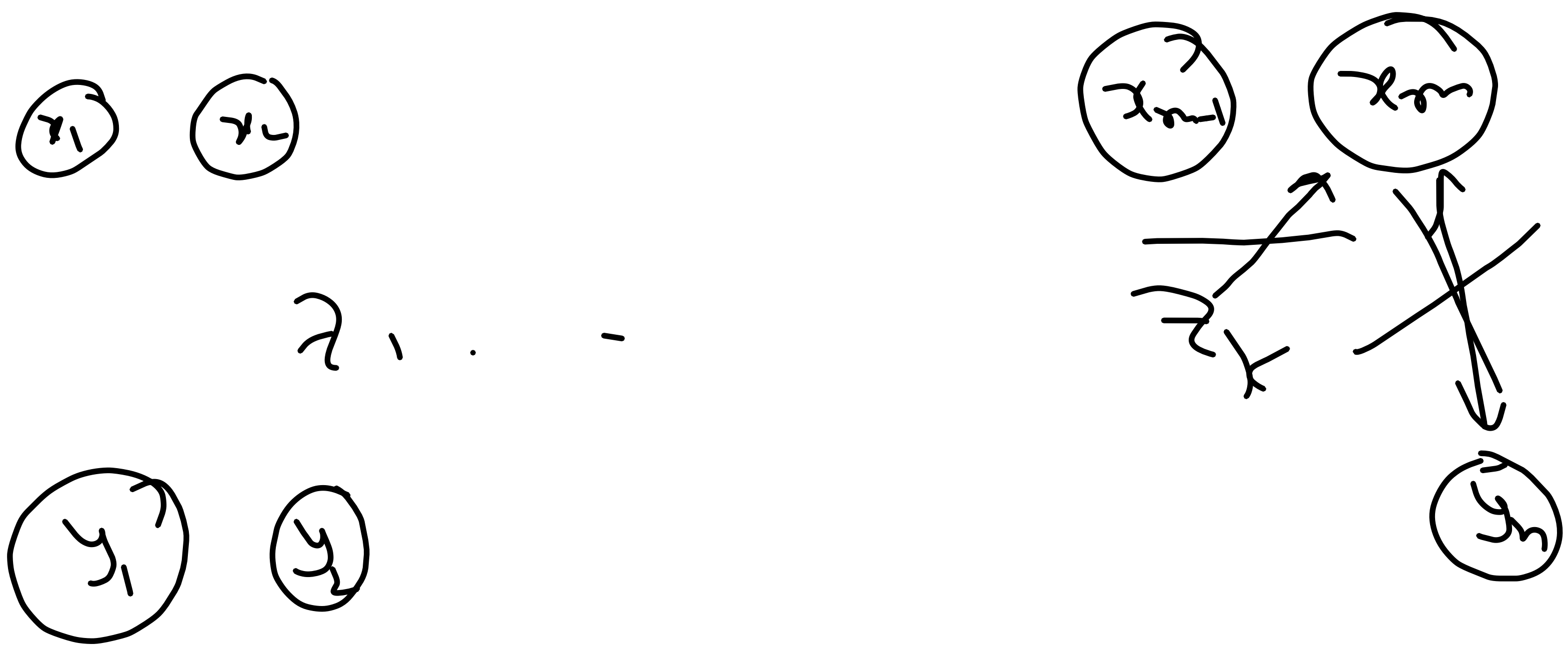
s.t. (i) Z is a subseq^n of both X & Y.

(ii) k is as large as possible.

$X = A\,B\,C\,B\,D\,A\,B$

$Y = B\,D\,C\,A\,B\,A$.

Let $Z = (z_1, \ldots, z_k)$ be a LCS of
$X = (x_1, \ldots, x_m)$ & $Y = (y_1, \ldots, y_n)$



Note.

(i) If $x_m = y_n$, then $z_k = x_m = y_n$ & $z_1 \ldots z_{k-1}$ is a LCS of $(x_1, \ldots x_{m-1})$ & $(y_1, \ldots, y_{n-1})$

(ii) If $x_m \neq y_n$, & $z_k \neq x_m$, then $z_1 \ldots z_k$ is a LCS of $x_1 \ldots x_{m-1}$ & $y_1 \ldots y_n$.

(iii) If $z_k \neq y_n$, $z_1 \ldots z_k$ is a LCS of $x_1 \ldots x_m$ & $y_1 \ldots y_{n-1}$

Subproblem: Find a LCS of $X_i = (x_1 \ldots x_i)$ & $Y_j = (y_1 \ldots y_j)$

Let $c[i,j]$ denote the length of a LCS of $(x_1 \ldots x_i)$ & $(y_1 \ldots y_j)$

$$c[i,j] = \begin{cases} 0 & \text{if } i=0 \text{ or } j=0 \\ 1 + \underline{c[i-1,j-1]} & \text{if } x_i = y_j \\ \text{Max}\left\{\underline{c[i-1,j]}, \underline{c[i,j-1]}\right\} \end{cases}$$

Dynamic Programming Sol^n for LCS

It stores $c[i,j]$ in a table $c[0..m, 0..n]$. This table is filled up in row-major order.

It also forms table $b[1..m, 1..n]$. Intuitively $b[i,j]$ will "point" to an entry in table that was chosen to compute $c[i,j]$

```
m ← length(X)
n ← length(Y)
for i ← 1 to m
    do c[i,0] ← 0
for j ← 0 to n
    do c[0,j] ← 0
for i ← 1 to m
    for j ← 1 to n
        do if x_i = y_j
            then c[i,j] ← 1 + c[i-1,j-1]
                 b[i,j] ← "↖"
            else if c[i-1,j] ≥ c[i,j-1]
                then c[i,j] ← c[i-1,j]
                     b[i,j] ← "↑"
                else c[i,j] ← c[i,j-1]
                     b[i,j] ← "←"
```

Matrix-chain Multiplication.

$A : p \times q$, $B = q \times r$. ; $C : r \times S$.

$A \times B$ needs pqr multiplications.

$ABC = ((AB) \cdot C) = (A \cdot (B \cdot C))$

D: s×t.

$$ABCD = ((AB).C).D$$

$$= A(B(CD))$$

$$= (AB)(CD).$$

$$= (A.(BC))D.$$

$$= (A((BC)D).)$$

A : 10 x 100

B : 100 x 5

C : 5 x 50

BC : 25,000
100 x 50

A : 10 x 100 → 50,000

75,000

$\underbrace{AB}_{10 \times 5}$ → 5000

C : 5 x 50 → 2500

7500

Def^n Given the product $A_1 \cdots A_n$ where A_i is $p_{i-1} \times p_i$, it is said to be fully parenthesized if either $n=1$ or it is a product of two parenthesized products surrounded by the parentheses (,)

		0	1	2	3	4	5	6
x		y	B	D	C	A	B	A
	0	0	0	0	0	0	0	0
A	1	0	0 ↑	0	0	1	1	1
B	2	0	1 ↖	1 ←	1 ←	1 ↑	2 ↖	2 ←
C		0						
B		0						
D		0						
A		0						
B		0						0

```
PRINT-LCS(b, X, i, j)

    if i = 0 or j = 0
            then return

        if b[i,j] ← ↖
            then PRINT-LCS(b, X, i-1, j-1)

                Print x_i
        else if b[i,j] ← ↑ then PRINT[b, X, i-1, j]
             else PRINT(b, X, i, j-1)
```

Ex Do away with the table b to obtain the longest common seq$^{\underline{n}}$ of X & Y.