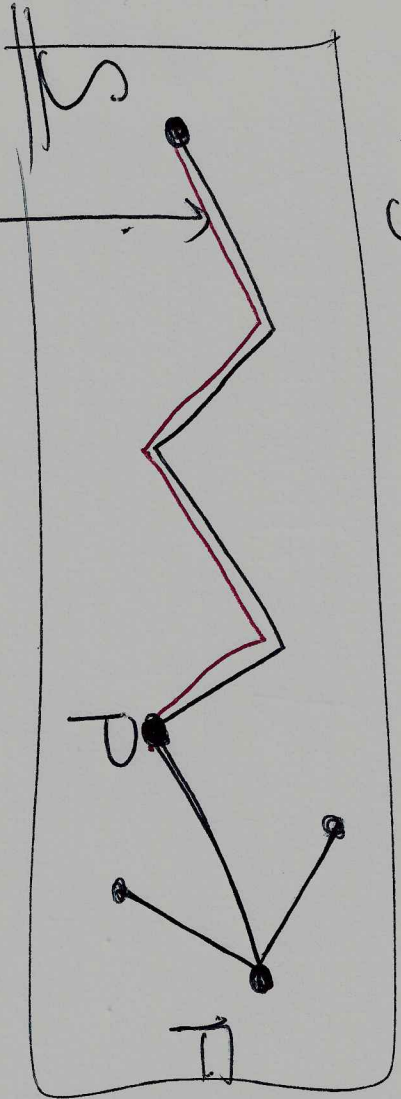


Data dependency analysis

Shortest path

Dyn. Pr

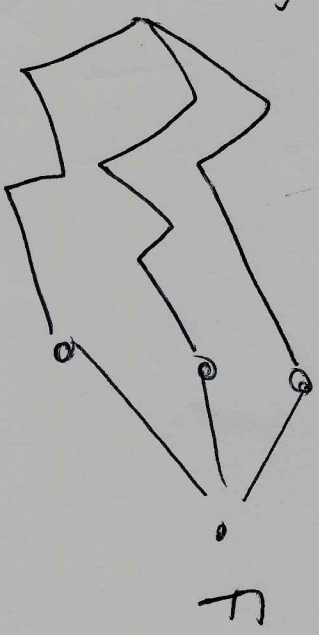


Shortest path \Rightarrow Last node before F is F's neighb.

Red path is SP shortest path from S to P

$$SP(F, k) = \min_{P \text{ neighbors of } F} \{ SP(P, k-1) + \text{len}(P, F) \}$$

k edges



min. $SP(s, P_m)$

$$SP(s, P_1)$$

$d[u][v]$
 // init ∞ , except $S \rightarrow 0$

for $k=1 \dots n$

for each u

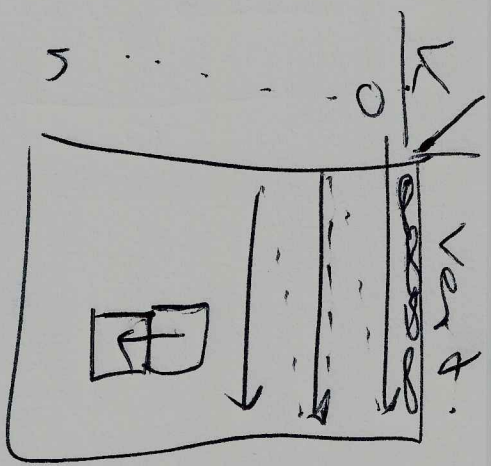
for each v

$alt = d[u-1][v] + len(u, v)$

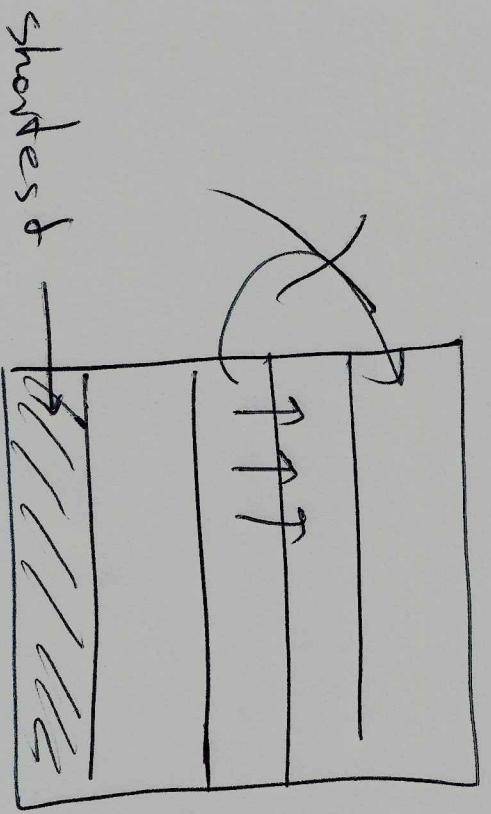
if ($d[u-1][v] > alt$)

$d[u][v] = alt$

else
 $d[u][v] = d[u-1][v]$

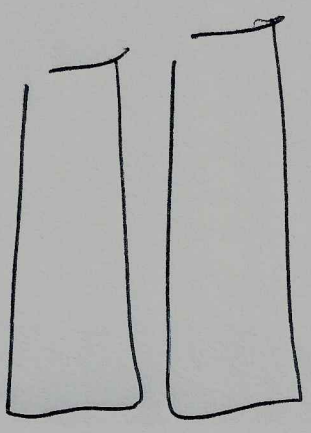


// alternative path with k edges $S \rightarrow v$



current

last (return)



Optimization - use 2 rows:

prev row } after iteration
current row }

prev = curr;

~~curr~~

$t[n+1] \leftarrow \text{all values.}$

Fig. 11

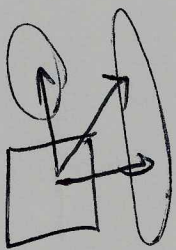
opt.

~~pp~~

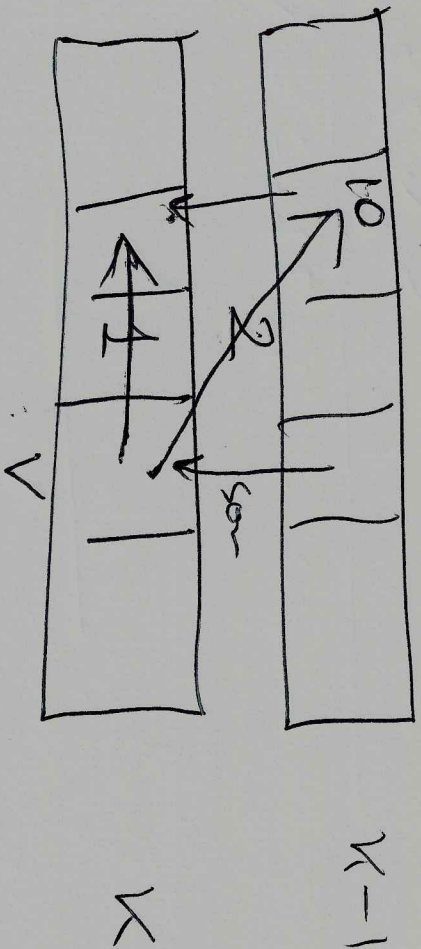
pp, p, c = p + pp;

pp = p, p = c, ...

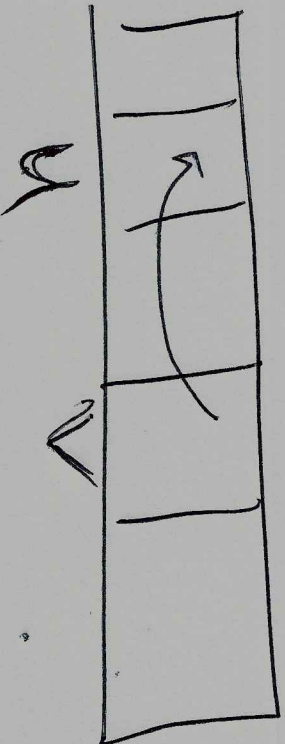
X
-3-



Back to Sh. Path



?



single arrays

$$\begin{cases} \text{alt} = d[u] + \text{len}(u, v) \\ \text{if } (d[v] > \text{alt}) \text{ } d[v] = \text{alt} \\ \text{else} \end{cases}$$

$$\text{alt} = d[u] + \text{len}(u, v)$$

$$\downarrow$$

$$\text{alt} = d[u] + \text{len}(u, v)$$

$d[u] := \{\infty\};$

$d[s] = 0;$

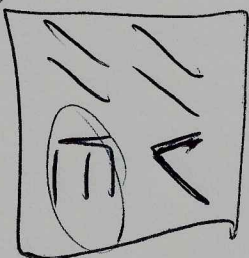
for $k = 1 \dots N$

for all edges in E ($u \rightarrow v$)

alt = $d[u] + \text{len}(u,v)$

if $(d[v] > \text{alt})$ $d[v] = \text{alt};$

Bellman-Ford!!! (SSSP)



Marshall-Floyd

-6-

APSP

for $k = 0 \dots n-1$

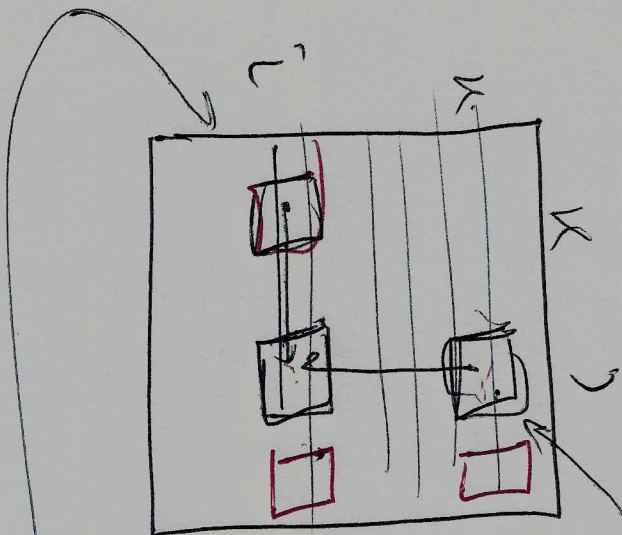
for $i = 0 \dots n-1$

for $j = 0 \dots n-1$

$$\text{if } i \neq j \text{ then } t[k][i][j] = \min$$

$$\left(t[k-1][i][k] + t[k-1][k][j], \right. \\ \left. t[k-1][i][j] + t[k-1][k][k], \right. \\ \left. t[k-1][k][i] + t[k-1][j][k] \right)$$

single 2-dim matrix



for $k=1 \dots n$

-7-

for $i=0 \dots n-1$

for $j=0 \dots n-1$

$d[i][j] = \min(d[i][j], d[i][n] + d[n][j])$

parallelize!

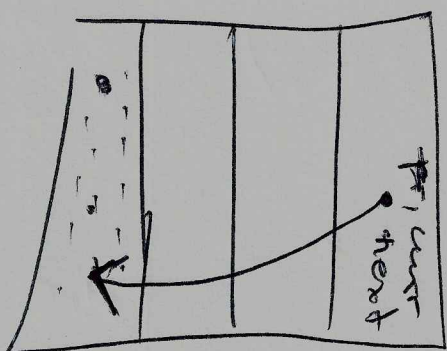
Parallelize?

no speed up!!

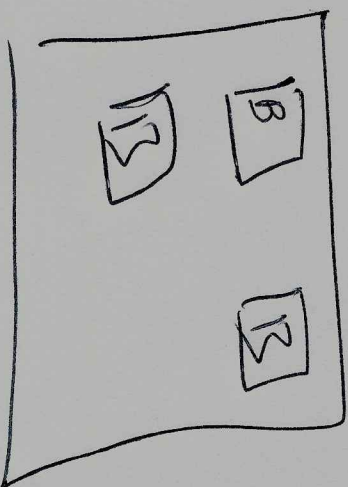
cache!

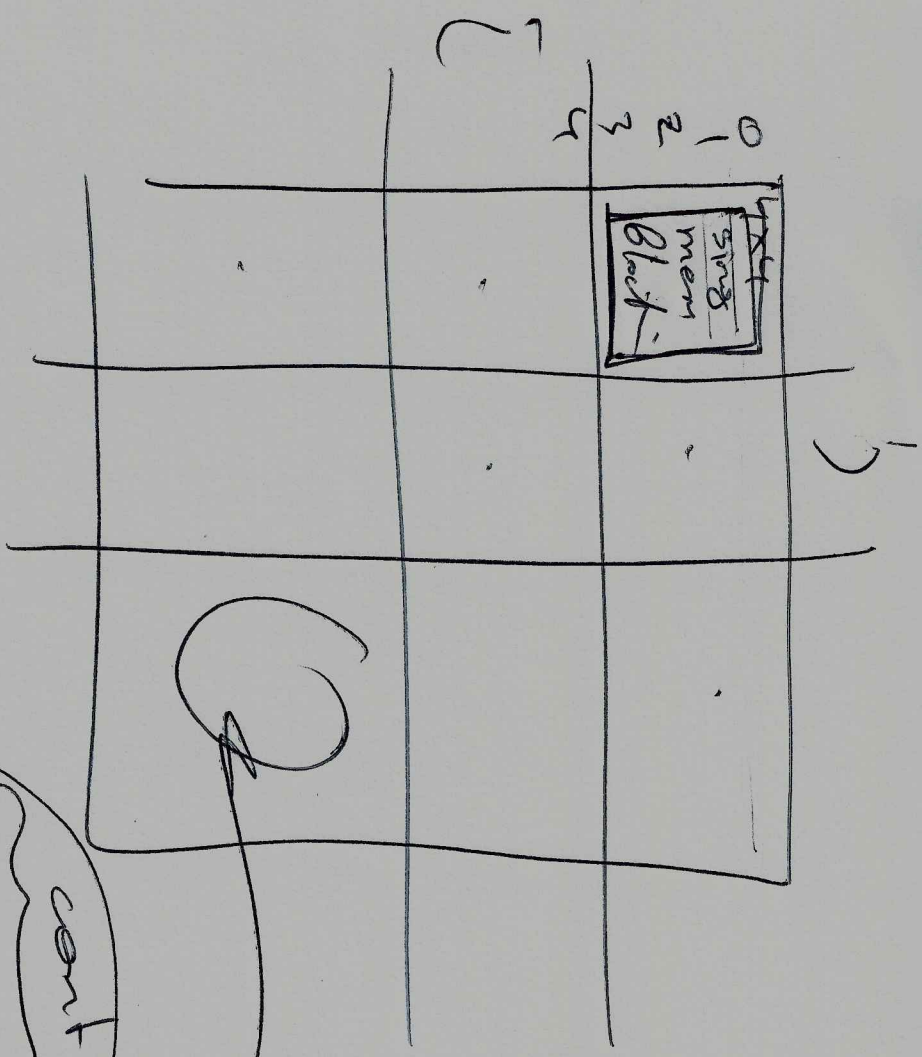
Ref. Locality is Bottleneck.

Blocked



T1
T2
T3
T4





int a [] [] []
whic
pos inside block

