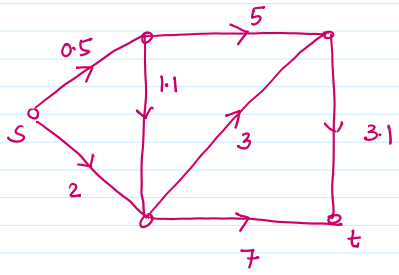
16) | Lecture 36.5-39.5 | Computing SSSP using Dijkstra Algorithm |

07 November 2024 10:16

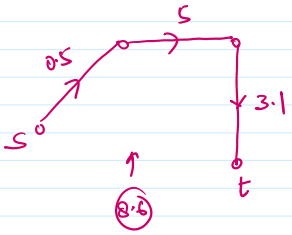
Single Source Shortest path

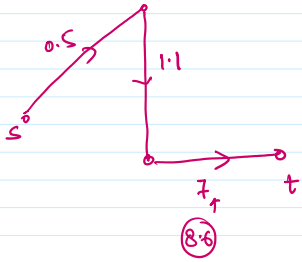
$G = (V, E)$ directed graph

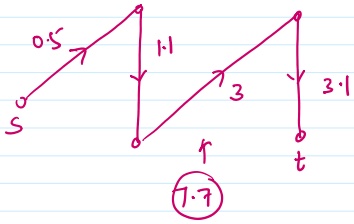
$l : E \to \mathbb{R}^+$



Aim is to find shortest path from s to t.

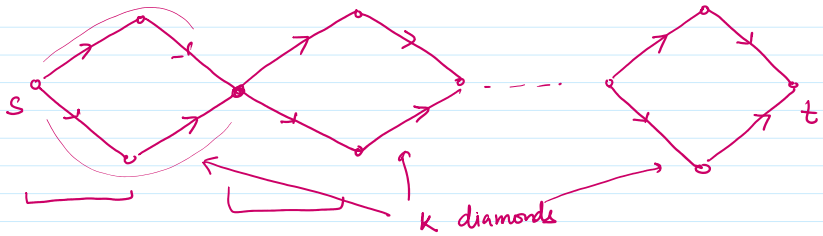






Naive Approach: Enumerate all path from s to t and report the min

Problem: # path can be very large — exponential in n.

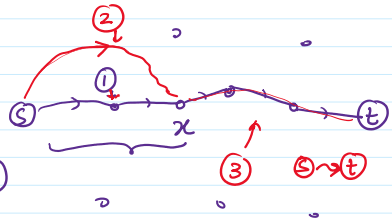


# of paths from s to t $= 2^K$ $\qquad n = 3K+1$

$= 2^{\left(\frac{n-1}{3}\right)}$

Property of shortest path

Claim Consider a shortest path from (s) to (t). And x is an arbitrary vertex on the path. Then (s) ⇝ (x) is shortest path from (s) to (x)



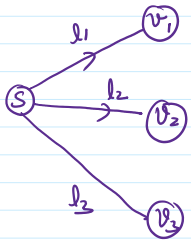
Proof Suppose not, let (2) is shorter than (1)

Consider path (2) + (x) ⇝ (t) should be shorter than (3), contradiction !!

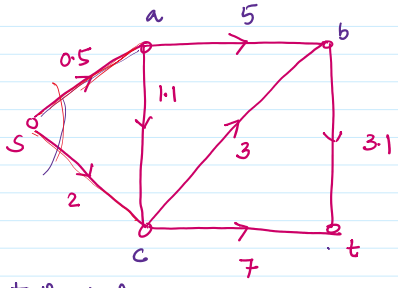
This is true for any arbitrary vertex sitting on shortest path from (s) to (t)

Single Source Shortest path problem (SSSP)

Aim is to compute shortest path from (s) to remaining vertex in graph



$l_1 < l_2 < l_3$



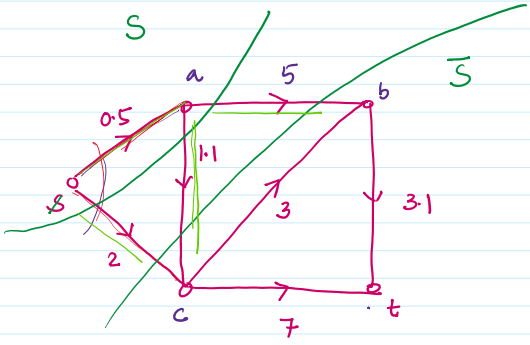
$\Rightarrow$ The length of shortest path from s to v_1 is l_1

$d[v_1] = l_1$, $d[v_2] \leq l_2$, $d[v_3] \leq l_3$ } upper bound

$d[v_1] = l_1$ ↑ exact

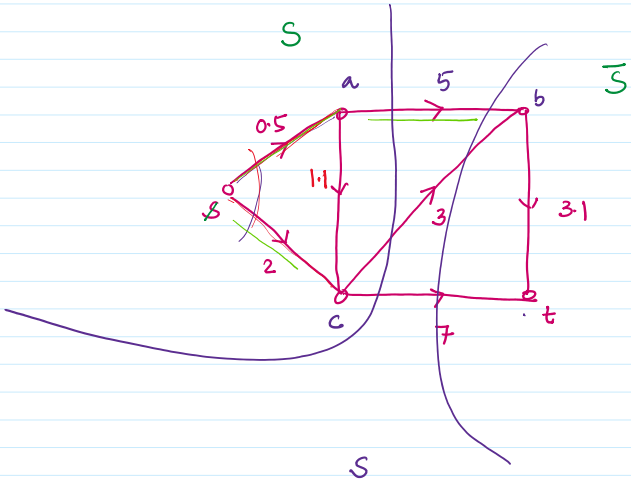
$d[v]$:= length of shortest path from s to v

$v \in V \setminus s$

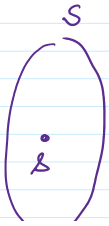


$\forall\, v \in S$ we know exact shortest path from s to v

$\forall\, u \in \bar{S}$ we know an upper bound on shortest path from s to u



Set of vertices to which we computed the shortest path from s



we have not found the shortest path. But we know an upper bound to it.

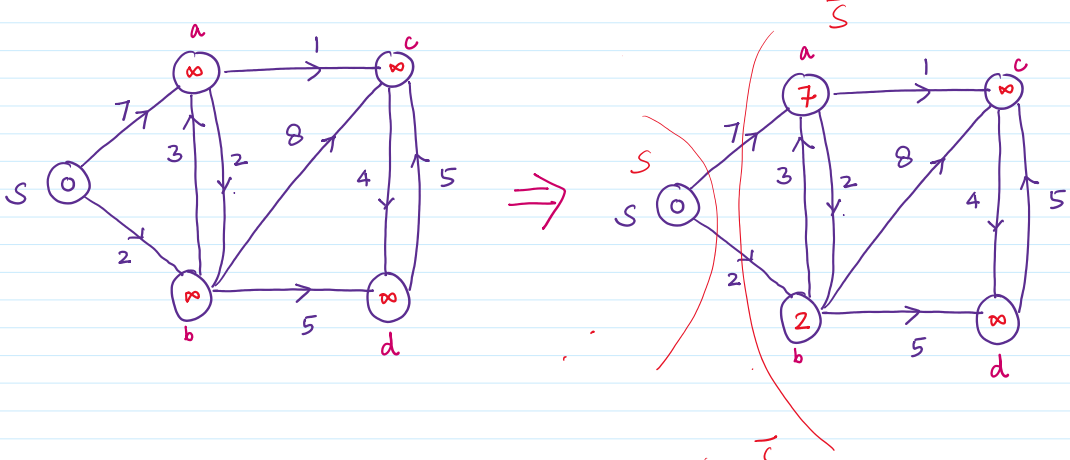
path from s

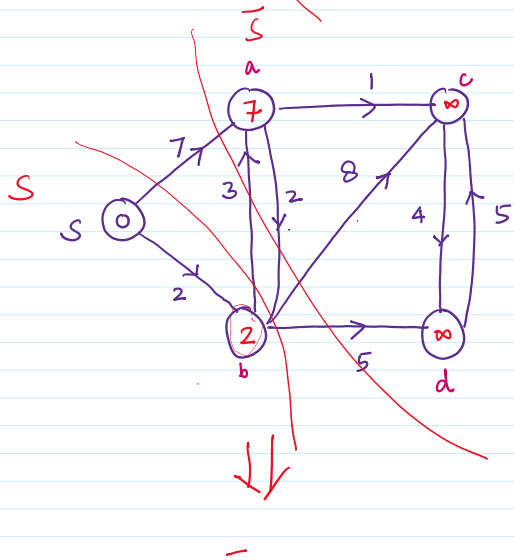
we know an upper bound to it.

In every step of algo. we compute the exact shortest distance of one more vertex in $\bar{S}$ & pull that vertex to S.

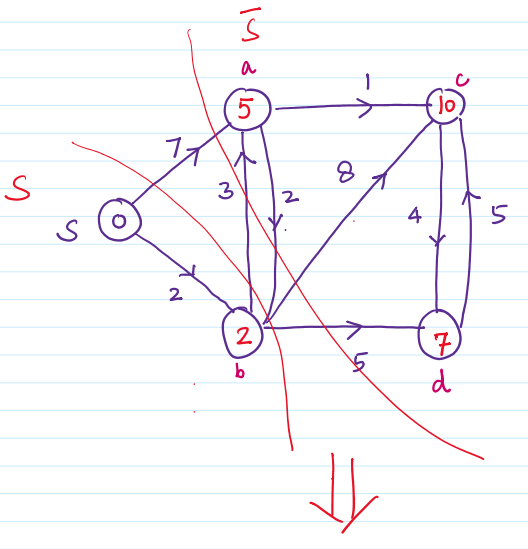
d[v] := length of shortest path from s to v

In each iteration, we pick vertex in $\bar{S}$ with min label, put it in set S and update label of vertices across partition.

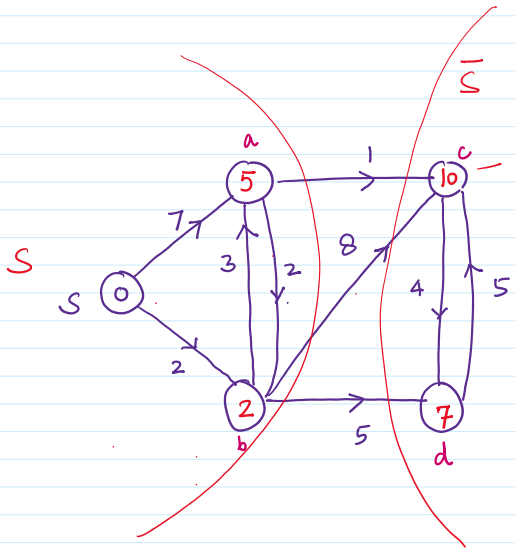


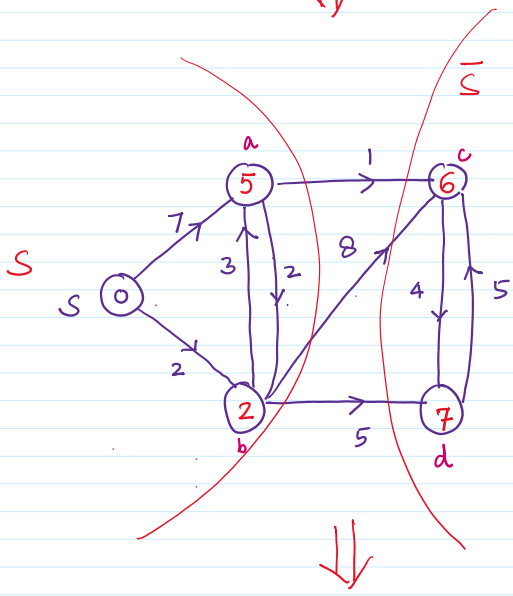


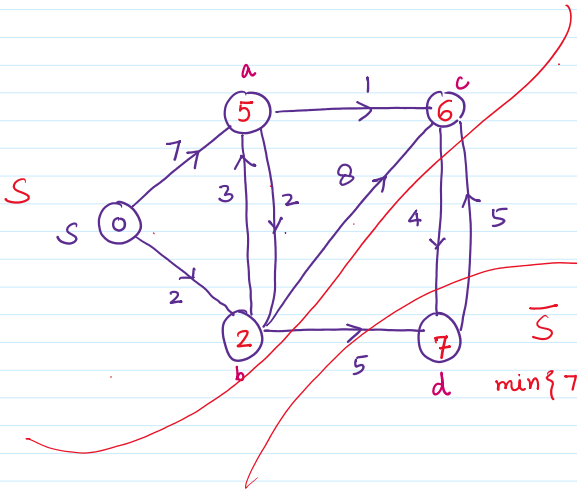
Picking vertex of min label



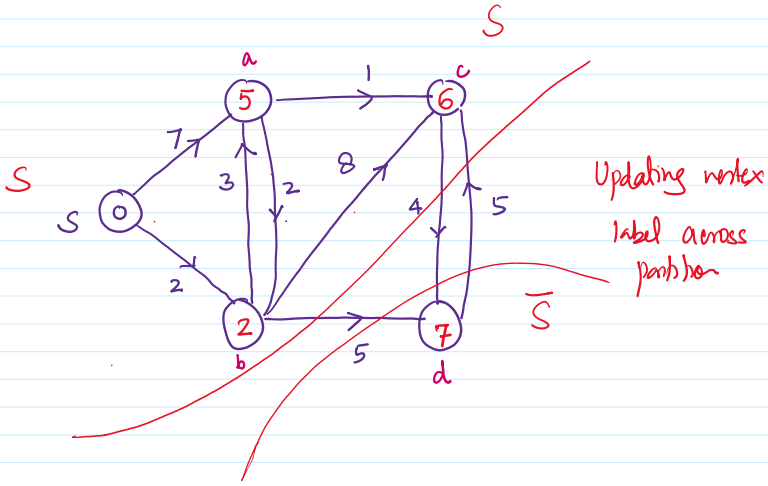
Updating label across the partition

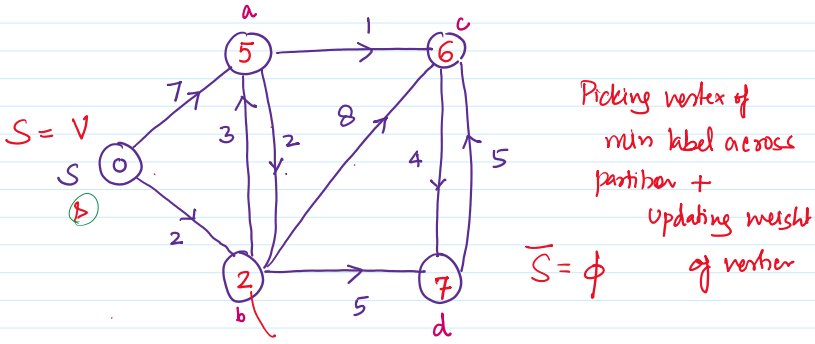

S
$\bar{S}$
a
b
c
d
5
10
2
7
0
1
7
3
2
8
4
5
2
5
min{10, 5+1}
Picking vertex of min label


S
$\bar{S}$
a
b
c
d
5
6
2
7
0
1
7
3
2
8
4
5
2
5
Updating label of vertex across partition


S
$\bar{S}$
a
b
c
d
5
6
2
7
0
1
7
3
2
8
4
5
2
5
Picking vertex of min label
min{7, 6+4}

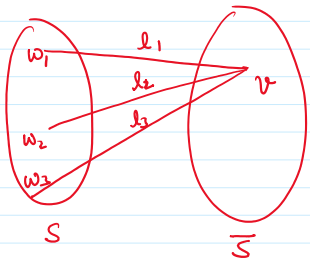
S





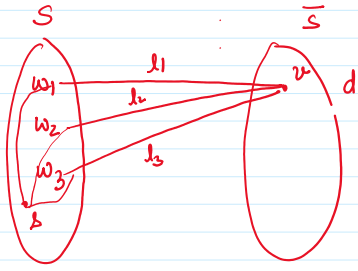
Once $S = V$ $d[v]$:= length of shortest path from s to v

Prims



label $[v]$ = min $\{l_1, l_2, l_3\}$

Djekstra



pick vertex from $\bar{S}$ that has min $d[v]$

Pseudocode

$d[s] = 0$, $\forall v \in V/s$, $d[v] = \infty$, H.insert(v, d[v])

/* maintain Heap for label of vertices in $\bar{S}$

while !H.empty() do /* until heap gets empty

{

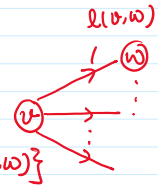
$v \leftarrow$ H.deletemin() ①

/* v is vertex with min label/priority in Heap

$l(v,w)$

/* v is vertex with min label/priority in Heap */

② For all w out adjacent to v do



③ d[w] ← min {d[w], d[v] + l(v,w)}

H.decreasepriority(w, d[w])

/* decrease priority of w

$$T(n,m) = \sum_{v \in V} (\log n + \deg(v) \log n)$$

$$= \sum_{v \in V} (1 + \deg(v)) \log n$$

$$= \Theta((n+m)\log n)$$

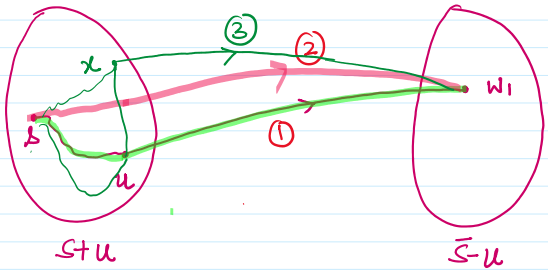
$$= O(m \log n)$$ [n is asymptotically smaller than m]

Proof of correctness:

let $d[u] \leq d[v] \quad \forall v \in \bar{S}$

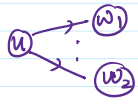
move u to S and update $d[w] = \min\{d[w], d[u] + l[u,w]\}$

d[w] always decreases/remain same $\forall w \in \bar{S}/u$



$d[w_1] \leftarrow \min\{d[w_1], d[u] + l[u,w_1]\}$

Argue about $d[w_1]$.



There are three possibilities of shortest path

① $s \rightsquigarrow u \rightarrow w_1$: shortest path from s to u and edge from u to w_1

② some other path from s to w_1 (without using u)

② Some other path from s to w_1 (without using u)

③ Path from s to u, then some other path from u to w_1

Case ① & ② are already covered by update rule.

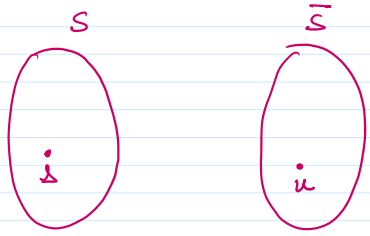
length of path $s \overset{*}{\rightarrow} u \overset{*}{\rightarrow} x \rightarrow w_1$ is not smaller than $s \overset{*}{\rightarrow} x \rightarrow w_1$

(path, edge)

Note that $s \rightsquigarrow x$ is shortest path from s to x, without including u
(so u just came in, x was already there)

But $d[w_1] \leq$ length of path $s \rightsquigarrow x \rightarrow w$

Proof by induction



$\forall x \in S$: $d[x]$ is the length of shortest path from s to x — (I)

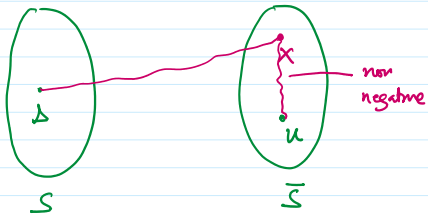
$\forall x \in \bar{S}$ $d[x]$ is the length of shortest path from s to x using only vertices of S — (II)

u is the vertex of smallest label in $\bar{S}$, and move it from $\bar{S}$ to S.

Then we need to show that $d[u]$ is the length of correct shortest path from s to u.

[After shifting u to S, we update the label of vertices in $\bar{S}$. Then need to show that Eqn (II) is true for remaining vertices in $\bar{S}$.

Claim I : $d[u]$ is the length of shortest path from s to u, where u is the vertex with shortest label.



Proof. Contradiction Suppose ∃ another shortest path via vertex $x \in \bar{S}$

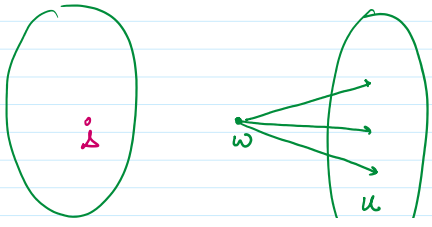
Entire shortest path from s to u is of weight $\geq d[x]$

$\geqslant d[u]$

O.w. we should have picked $x \in \bar{S}$ and shift to S instead of u

Contradiction

How to compute shortest path



If $d[u] = d[w] + l[w, u]$, then shortest path from s to u (using only vertices in S & w) has w preceding u

shortest path from s to u (using only vertices in S ∪ {w}) has w preceding u

w is moved to S

$d[u] \leftarrow \min\{d[u], d[w] + l(w,u)\}$

∀ each vertex we maintain predecessor information

```
if d[u] > d[w] + l[w,u]
then d[u] = d[w] + l[w,u]
     pred[u] = w
```

How to compute shortest path s to v using pred[] array

```
x = v
while (pred[x] != null) then
[
    print (x)
    x = pred[x]
]
```

/* print shortest path in reverse order