

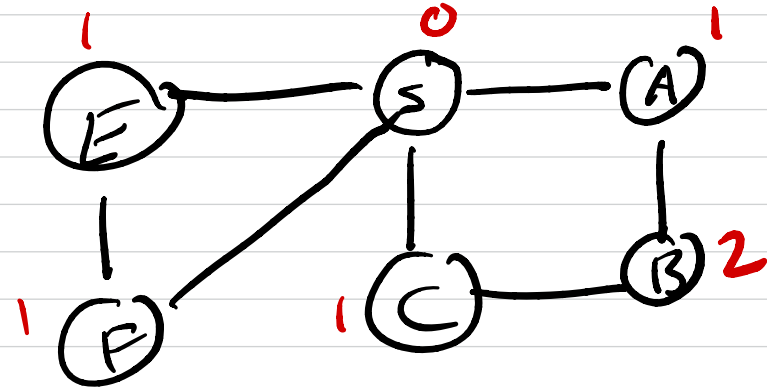
Paths in

Graphs

Single-source shortest paths (SSSP)

Input: Graph G , "source" vertex $s \in V$

Output: $\forall u \in V$, $d(s, u)$ = length of shortest path from s to u



Unweighted: all edges length 1

Breadth-first search

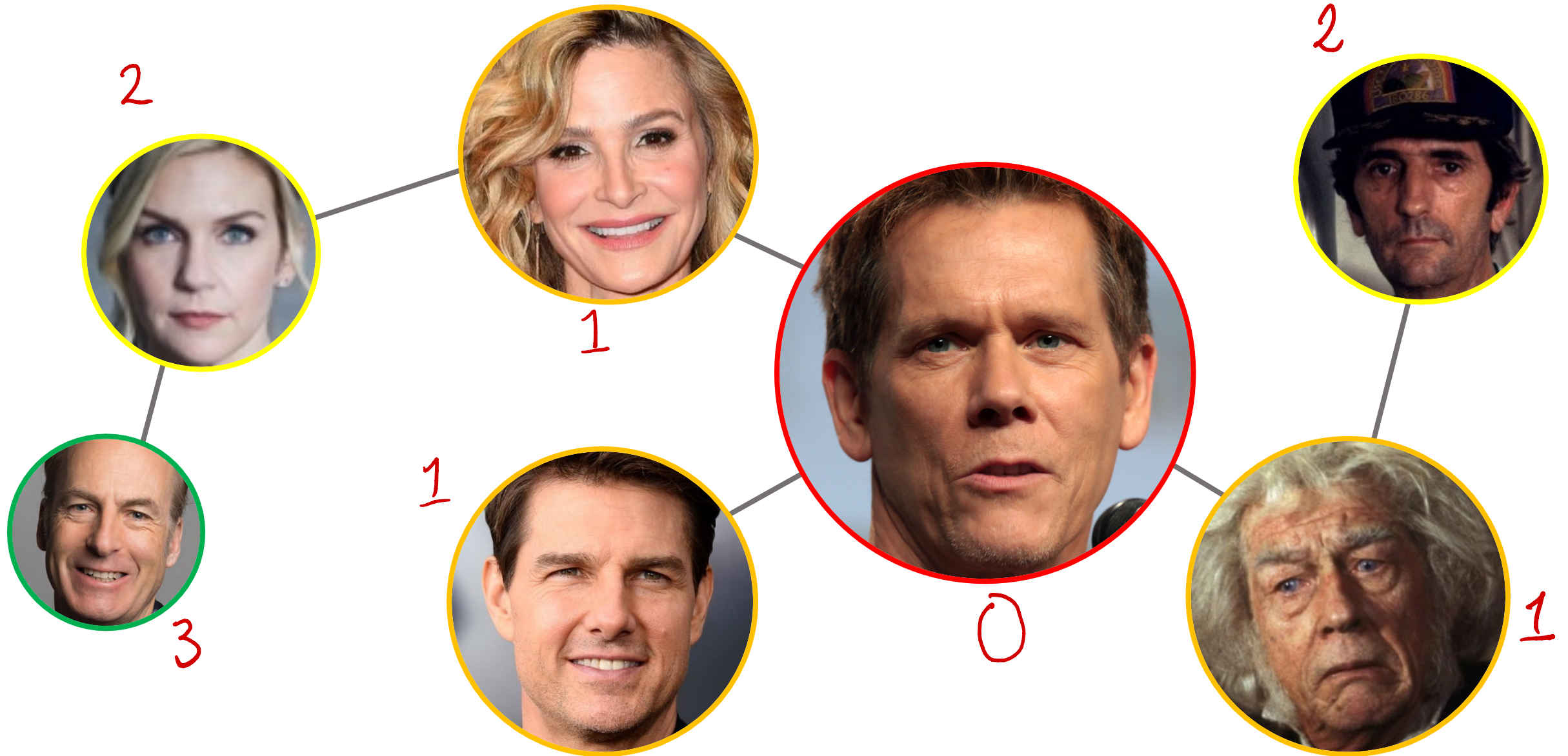
Positive lengths: $\ell: E \rightarrow \{1, 2, 3, \dots\}$

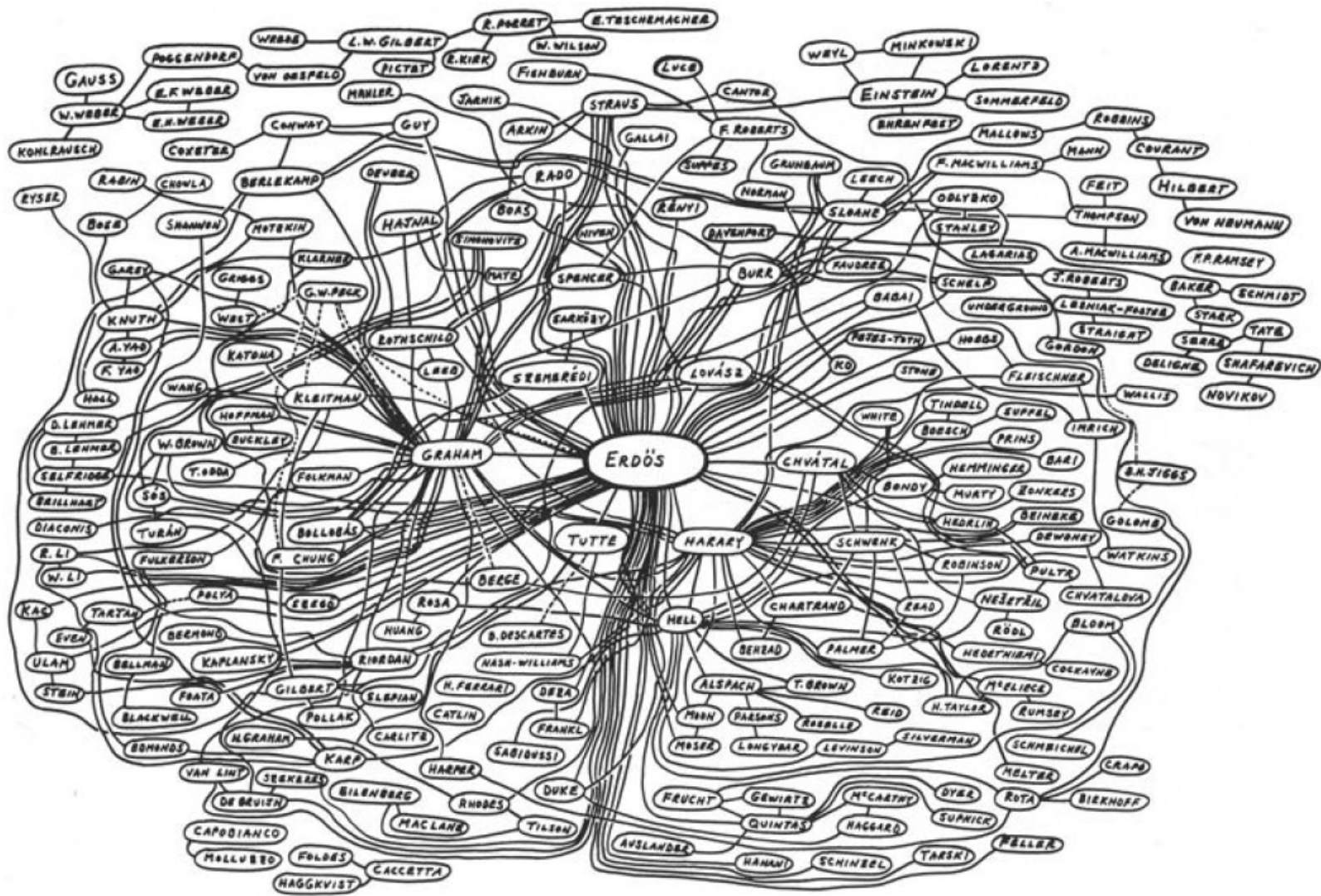
Dijkstra

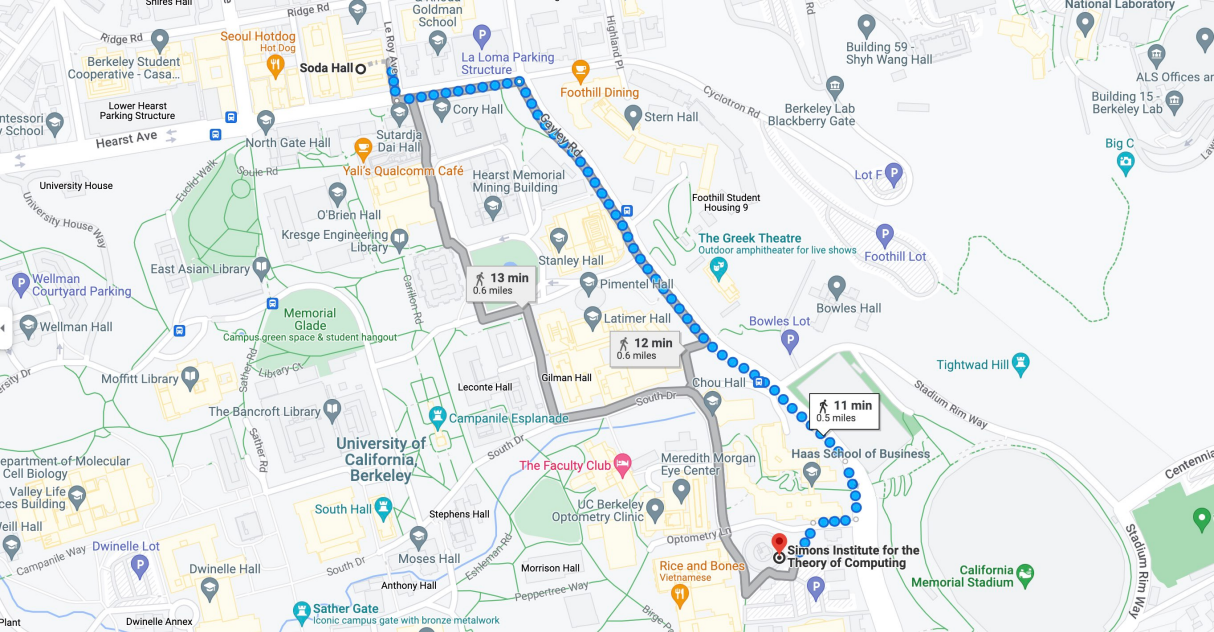
Arbitrary length edges

Bellman-Ford

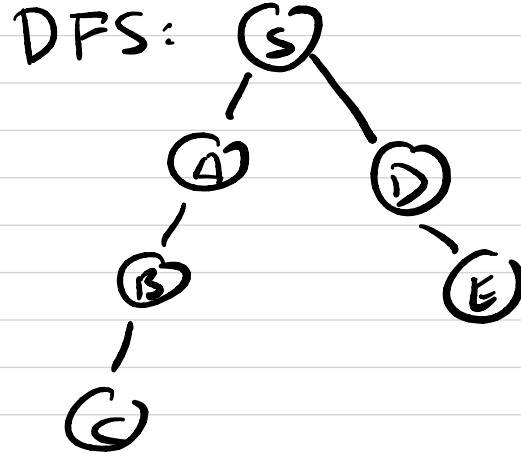
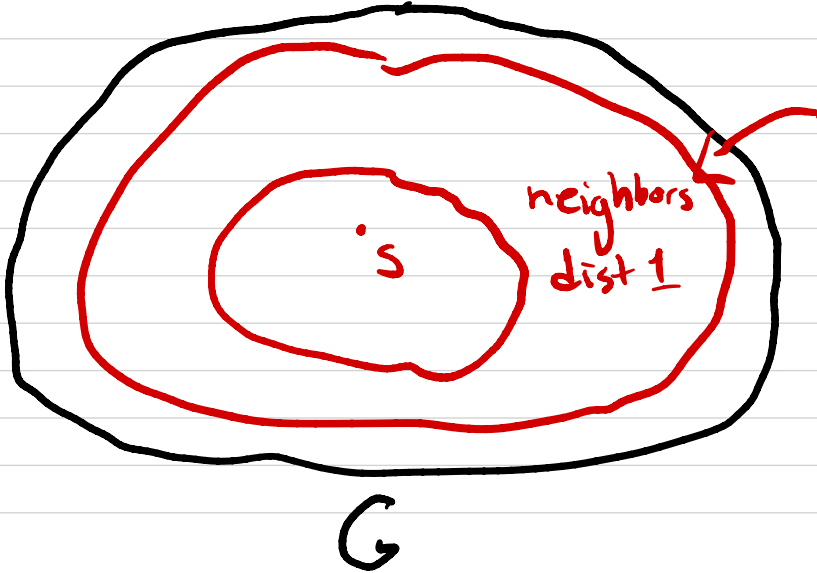
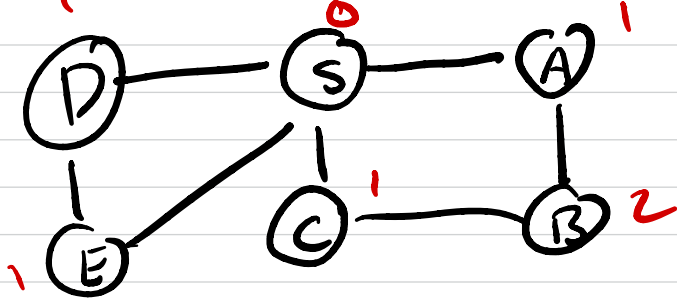
Application: Kevin Bacon number



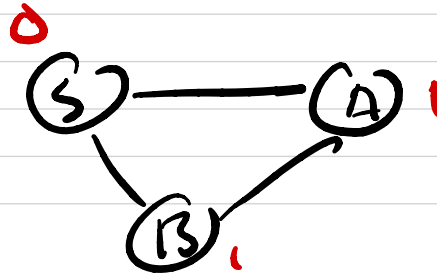




Unweighted graphs



neighbors of neighbors
(have not yet seen)
dist 2



Breadth-first search

bfs (G, s)

$\text{dist}[s] = 0$

$\forall u \neq s, \text{dist}[u] = \infty$

$Q = \{s\}$ (queue containing s)

while Q is not empty

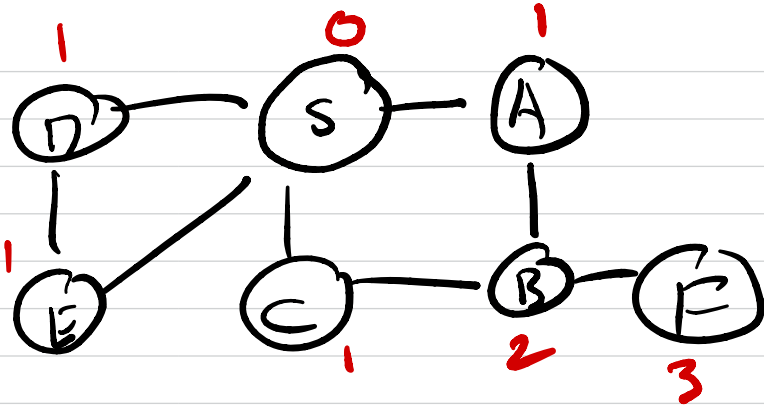
$u = \text{dequeue}(Q)$

for all v s.t. $(u, v) \in E$

if $\text{dist}[v] = \infty$

enqueue(Q, v)

$\text{dist}[v] = \text{dist}[u] + 1$



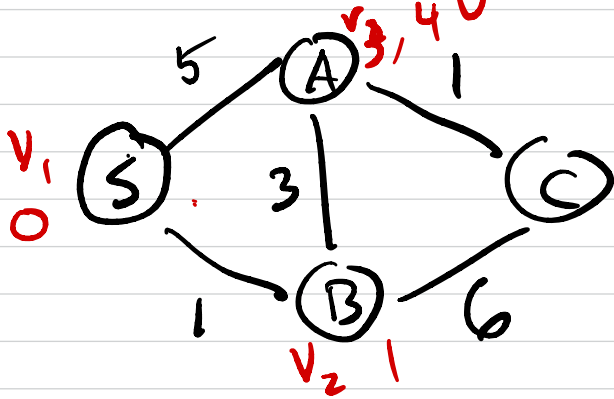
$Q: \cancel{S} \checkmark \checkmark \checkmark \checkmark \checkmark \checkmark \checkmark$

Runtime: $O(n+m)$ time

linear, same as D_{FS}

DFS is just BFS w/ stack

Positive lengths



Dijkstra's algorithm

1. Compute $v_1 =$ closest vertex to s
and $d(s, v_1)$
2. Compute $v_2 = 2^{\text{nd}}$ closest to s
and $d(s, v_2)$
3. " v_3 3^{rd} " "

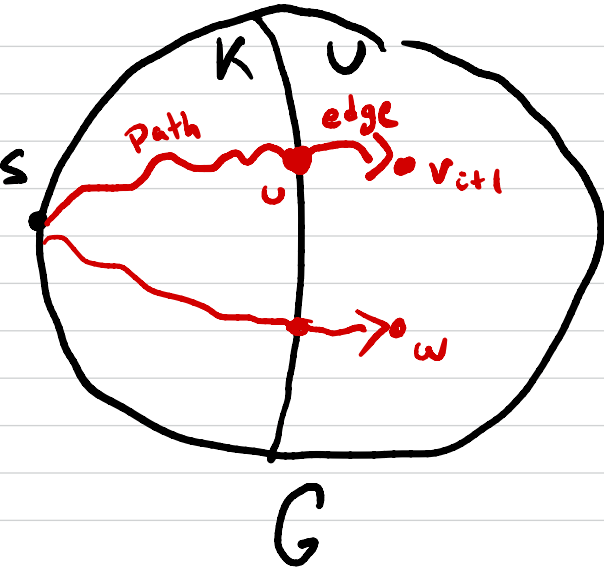
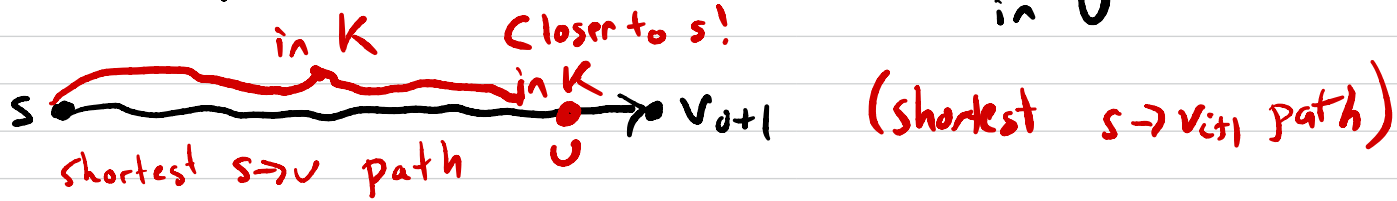
$K =$ the set of "known" vertices
at some step

$$= \{v_1, \dots, v_i\}$$

$U =$ "unknown" vertices $U = V \setminus K$

Q: Given $K = \{v_1, \dots, v_k\}$.

How to compute v_{i+1} ? = closest vertex not in K in U

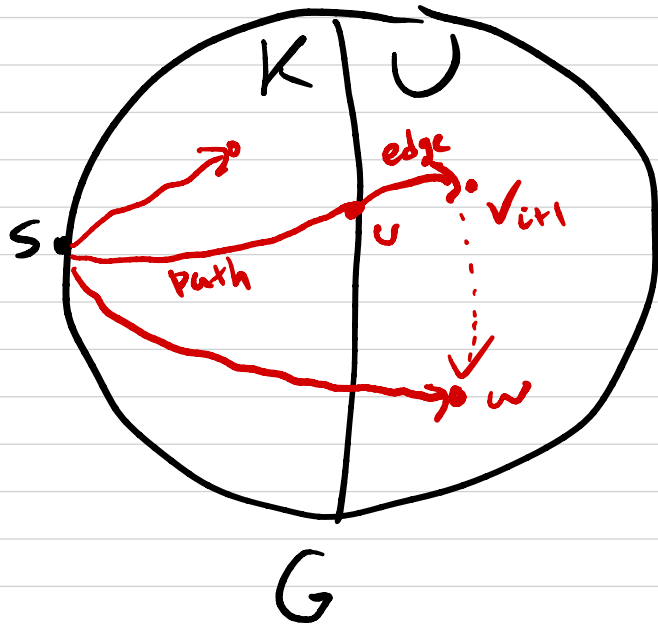


v_{i+1} = endpoint of the shortest path of this form

$$\text{dist}[v_{i+1}] = \min_{u \in K} \left\{ \text{dist}[u] + l(u, v_{i+1}) \right\}$$

Dijkstra has $\text{dist}[v]$ for each $v \in V$

$$\text{dist}[v] = \begin{cases} d(s, v) & \text{if } v \in K \\ \min_{u \in K} \text{dist}[u] + l(u, v) & \text{if } v \notin K \end{cases}$$



After adding v_{i+1} to K
If $(v_{i+1}, w) \in E$

$$\left[\text{dist}[w] = \min \left\{ \text{dist}[w], \text{dist}[v_{i+1}] + l(v_{i+1}, w) \right\} \right]$$

update(v_{i+1}, w)

Dijkstra (G, l, s)

$$\text{dist}[s] = 0$$

$$\forall u \neq s, \text{dist}[u] = \infty$$

$$U = V \quad (\text{insert}(u, \text{dist}[u]) \quad \forall u)$$

while U is not empty

Choose $u \in U$ with minimum

Remove u from U . $\text{dist}[u]$

$(u = \text{DeleteMin}(U))$

for each v s.t. $(u, v) \in E$

$$\text{dist}[v] = \min(\text{dist}[v],$$

$$\text{dist}[u] + l(u, v))$$

$\text{DecreaseKey}(v, \text{dist}[v])$

Priority queue

Contains a set of
(element, key) pairs
integer

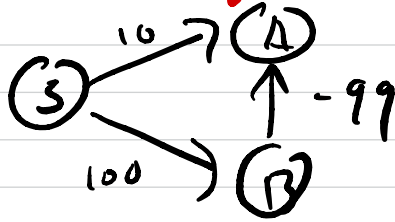
- Insert (elem, key)
- Decrease Key (elem, key)
(Replaces elem's old key
w/ new key)
- DeleteMin()

Dijkstra's runtime

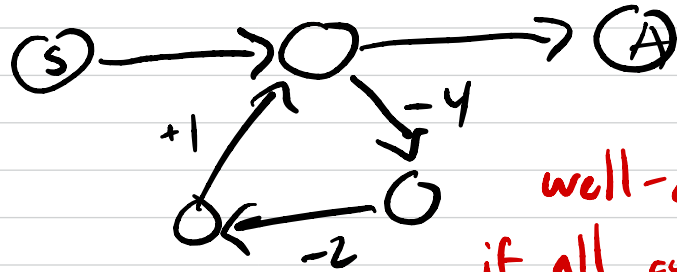
Insert n times
Delete Min n times
Decrease Key m times

Implementation	Insert	Del Min	Decrease	Total
array	$O(1)$	$O(n)$	$O(1)$	$O(n^2 + m) = O(n^2)$
binary heap	$\log(n)$	$\log(n)$	$\log(n)$	$O((n+m) \log(n))$
Fibonacci heap	$O(1)$	$O(\log(n))$	$O(1)$	$O(n \log(n) + m)$

Negative weights



Does it make sense?



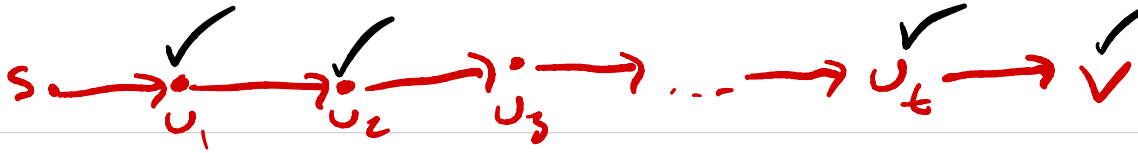
well-defined
if all cycles
have positive
lengths

update(u, v) $u \rightarrow v$

$$\text{dist}[v] = \min \left\{ \text{dist}[v], \text{dist}[u] + l(u, v) \right\}$$

① Update is "safe" : $\text{dist}[v] \geq d(s, v)$

② Suppose shortest $s \rightarrow v$ path is $s \rightarrow \dots \rightarrow u \rightarrow v$
and $\text{dist}[u] = d(s, u)$. Then after update, $\text{dist}[v] = d(s, v)$



→ update(s, u)

→ update(u_1, u_2)

⋮

→ update(u_{t-1}, u_t)

→ update(u_t, v)

Bellman-Ford(G, l, s)

For $i = 1 \dots N-1$

Update all the edges

(for all $(u, v) \in E$,
update(u, v))

Runtime: $O(nm)$