Recitation 15

SHORTEST PATHS

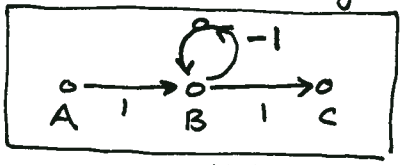
unweighted graphs : 2 way BFS = shortest path.
every edge counts equally - shortest = min # nodes.

weighted graphs : for each edge associate a number $w(u\,v)$
weighted paths : $$W(p) = \sum_{\substack{\text{edge} \\ (u,v) \\ \in p}} \left[W(u\,v) \right]$$

- commutative
- $W(p \to x) = W(p) + W(p_{end} \to x)$
- $W(u \rightsquigarrow x) + W(x \rightsquigarrow v) = W(u \rightsquigarrow x \rightsquigarrow v)$
- linear accumulation

shortest path $\delta(u, v)$ = $\begin{cases} \min [W(p) : u \overset{p}{\rightsquigarrow} v & \text{path from } u \to v. \\ \infty & \text{no path exists} \end{cases}$

- corner case : negative weight cycle

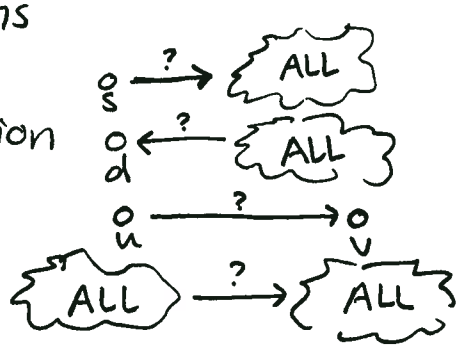


possible responses
- $\delta(A,C) = -\infty$ $A \to B \to B \to \cdots \to B \to C$
- simple paths : no loops allowed
  - hard / expensive to enforce since ALL min paths algs capitalize on locality.

- longest ? pset 5.

Shortest paths problems

- single source
- single destination
- single pair
- all pairs



reverse edges.

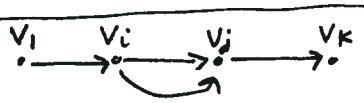
brute force : can be done faster though.

optimal substructure

$p = (v_1\ v_2\ \cdots\ v_k)$ = shortest path from v_1 to v_k

$p_{ij} = (v_i\ \cdots\ v_j)$ = subpath from v_i to v_j

$\Rightarrow p_{ij}$ is a shortest path.



proof: assume p'_{ij} and $w(p'_{ij}) < w(p_{ij})$

subst. p'_{ij} in p : then $w(p') = w(v_1 v_i) + w(p'_{ij}) + w(v_j v_k)$

$w(p) = w(v_1 v_i) + w(p_{ij}) + w(v_j v_k)$

$w(p') < w(p)$ contradiction since p shortest.

cycles? can a shortest path contain a cycle? No cut out cycle = shorter

using optimal substructure to represent paths.

$p_{k+1} = (x_1 \cdots x_k\ x_{k+1})$ shortest path. to x_{k+1}

$p_k = (x_1 \cdots x_k)$ shortest path to x_k

....

$p_1 = (x_1)$ shortest path to x_1

So only need to store the predecessor on the shortest path!

$O(v)$ extra storage.

- $\pi[v] = pred.$

forms a shortest paths tree — why tree? b/c we choose ONE min path

subgraph $G'_s = (V'\ E')$ $V' \subseteq V$ $E' \subseteq E$ ← tree rooted @ s.

↑ reachable from s.

unique path from $s \to v \in G'_s$ = min path in G from s to v.

Relaxation

- maintain a shortest path estimate $d[v]$ – current min. dist.
- initialize $d[v] = \infty$ $\pi[v] = none$
  $d[s] = 0$ $\pi[s] = none.$
- relaxing can we improve any $d[v]$?
  - algs supply when to relax a particular edge.
  - will never relax a shortest path

```
Relax(u v w)
if d[v] > d[u] + w(u v):
    d[v] = "
    π[v] = u
```

- properties CLRS p587

RUBIKs code

Interface — representations

position : a particular state of a cube
I : the identity / solved position.
move : represent a possible move
F , Fi : rotate front face clock/cclock
L , Li :
R , Ri :

} quarter twists = tuple of these.

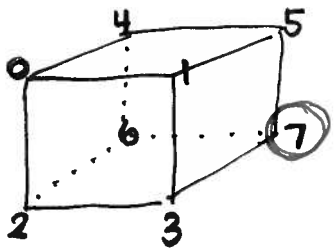
operations

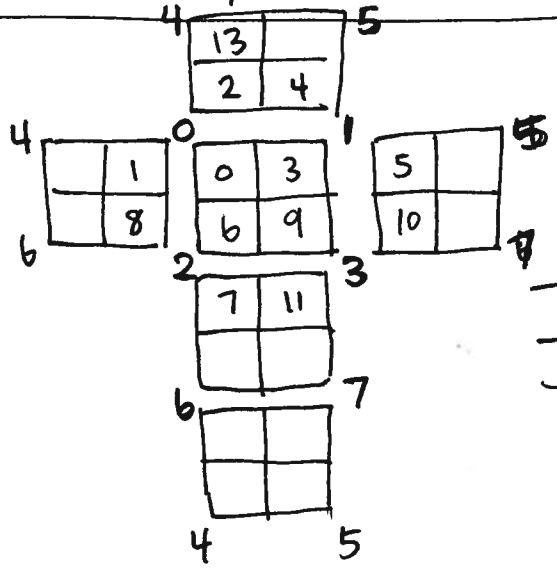
perm_apply (move , position) → new state of cube
perm_inverse (move) → inverse move
equals (move1, move2)
equals (pos1, pos2)

if computing this is ALL you need to know!

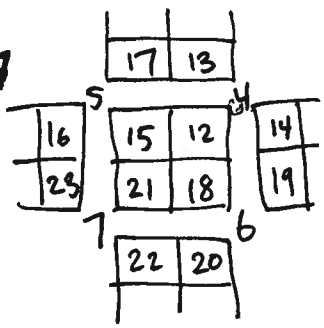
‼ simple
‼ but what about my pocket cube.

best: a user interface that masks the representation (10 pnts)

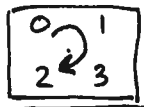
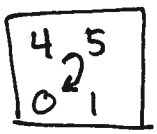
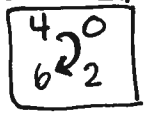




precedence
front
back



moves:

F 
U 
L 

code swaps yellow + blue ! sorry. batch soon.

move: L