D

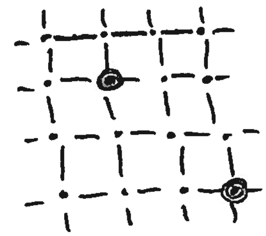
Recitation 14

- BFS + Path Finding
- DFS + edge classification
- topological sort, DAGs

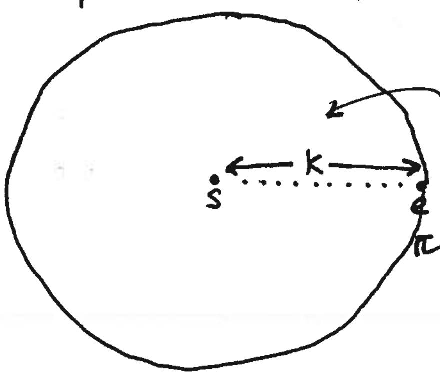
— Path Finding with Breadth First Search

- BFS finds shortest paths
- time : if shortest path = k nodes long
  BFS searches all k-reachable nodes.
- space : each step adds (degree) nodes to Q.

example



K = 5
degree = 4.
time = K^2 area of circ.
space = $O(K)$ perim of circ



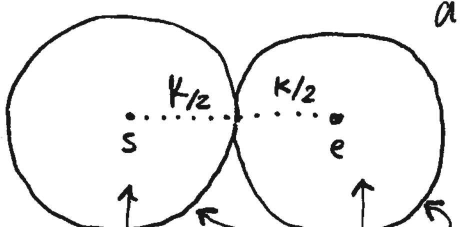
searched region = $\Theta(k^2)$

items in Queue = $\Theta(K)$

constant density of nodes / edges.

— Improvement: 2-way BFS

- idea : simultaneously search forward from start and backward from end.



searched area

items in Queue

	one way	two way
$\lvert R \rvert$	πk^2	$2\pi(\frac{k}{2})^2 = \frac{\pi k^2}{2}$
$\lvert Q \rvert$	$2\pi k$	$2\pi \frac{k}{2} \cdot 2 = 2\pi k.$

- even better improvement w/ large branching factors / less overlap.
  - tree w/ branching factor b.

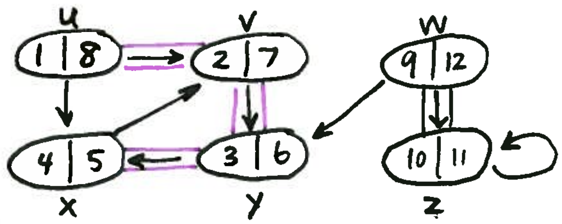
$$R = b^k \qquad\qquad 2b^{k/2} = 2(\sqrt{b})^k$$
$$\longrightarrow$$
$$Q = b^{k-1} \qquad\qquad 2b^{(\frac{k}{2}-1)} \approx 2(\sqrt{b})^{k-1}$$

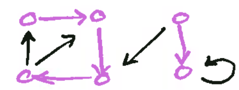
- still guarantees shortest path.

— shortest path proof

Edge classification with Depth First Search

- idea: perform DFS starting @ arbitrary nodes until all nodes traversed
  - for each node keep track of
    - time node first visited
    - time node completed



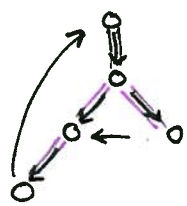


- "parenthesis structure": open paren @ start, close paren @ finish.

  1 2 3 4 5 6 7 8 9 10 11 12

  (((( )))) (( ))

ex 2:



→ (((( )) ( )))

- always properly matched no)()(or))((
- A inside B means A is a descendant of B
- A disjoint B means no familial relation: different subtrees.

edges
- tree : —
- back : desc → ancest
- forward : ancest → desc
- cross : no familial
- loop : —

Topological Sort

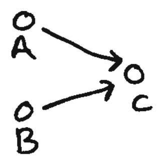
- Directed Acyclic Graph
  - frequently used to define a constraint graph.

ex: course requirements
computations

problem: determine a listing of nodes such ~~that~~ if $(i\ j) \in E$ then i listed before j. = topological sort.

3)

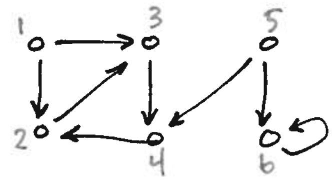
- solution NOT unique



⇒ A B C or B A C

- one answer : use completed times from DFS ⇒ $O(V+E)$ will examine every vertex & edge.
  - why not use visited times?

- can we use BFS ? NO



- progressively remove nodes w/ degree 0.
  - scan graph for nodes w/ indegree 0 → put in set I.
  - while I not empty
    - remove arbitrary node from I → n
    - for all n's children : decrease indegree by 1.
      - if indegree = 0 add to I
    - add n to end ~~of~~ topo sort list.

? cycles.

⇒ $O(V+E)$

- other ideas?
  - reverse : remove outdegree 0 + put @ end.

---

extras : compute diameter of a graph
define connected component
strongly connected component
singly connected ~~exactly~~ at most one path $u \rightsquigarrow v$ for all u, v