CS3000
6/3 - tue.

Admin

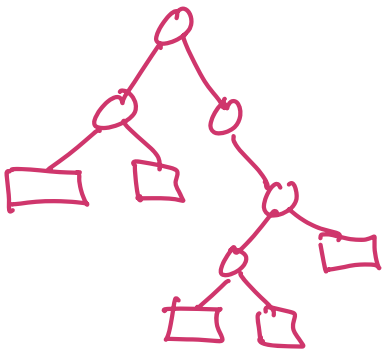
- HW4 due 6/5 9pm
- Exam #2 6/12 in class
- recitation today!

Agenda

1. Graph Overview
2. Breadth First Search
3. BFS Proof!!

0. Huffman Review

- prefix free → the code for a character is not a prefix for any other code
  - ex) B - 101, no other code starts with 101
- greedy choice? → extract min 2 lowest freq characters
- valid: prefix free, encode, decode
- optimal soln: encode using the fewest # of bits



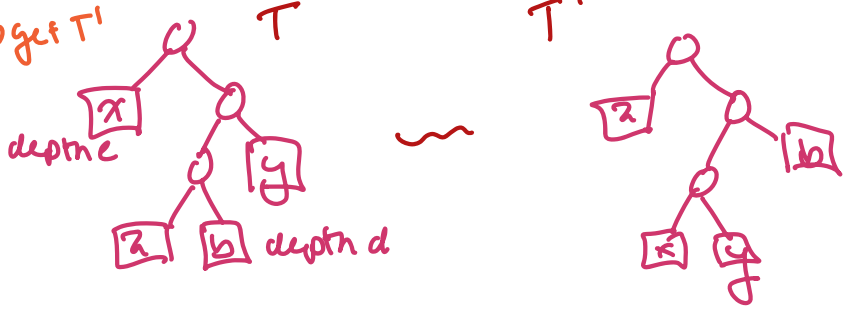
$$B(T) = \text{total bits} = \sum_{c \in C} c.freq * c.depth$$

T - optimal soln
a, b - chars at max depth in T
x, y - chars w/ lowest freqs (greedy choice =!)

Inter. step: swap x, a and y, to get T'

$B(T) - B(T') =$
$(x.freq * e + a.freq * d)$
$- (x.freq * d + a.freq * e)$
$= (a.freq - x.freq)(d - e)$

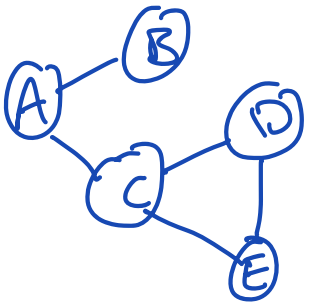
≥ 0 because
$a.freq \geq x.freq \geq$
$d \geq e$



swap x, y with a, b
$B(T) \geq B(T')$ — argue

1. Graph Overview

;; can we do better? ;; naive → data structure → D+C → DP → greedy → graphs!



- 5 vertices
- connected
  - ↳ path from any vertex to any other vertex
- cycle
  - ↳ path from a vertex to itself
- unweighted
- simple
  - ↳ no self loops, no multi edges

- undirected
- deg(A) = 2
  - ↳ # edges incident on A
- deg(graph) = 10
- no default root
  - ↳ specify a source vertex
- cycle
  - ↳ does not repeat edges

C,D,E,C cycle!
A,B,A not a cycle!

path from u to u'
sequence of vertices $\langle v_0, v_1, v_2, \dots, v_k \rangle$
where $u = v_0$ $u' = v_k$
and v_{i-1}, v_i is an edges

cycle path where $v_0 = v_k$
and there is at least one edge

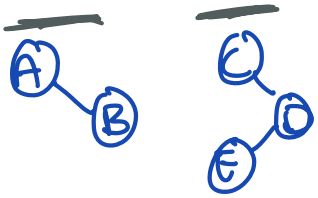
strongly connected?
- no !!
- path btw every pair of vertices

In pseudocode
$G = (V, E)$

graph, vertex w/o edges ok!

edge w/o vertices doesn't exist

↳ connected components

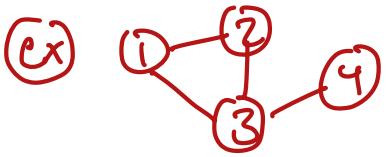


unconnected

one graph

Algo(G)
↳ graph

How to represent G?
- adjacency list
- adjacency matrix



for $v \in G.V$...

Adj list
- every vertex has linked list of neighbors

1: 2, 3
2: 1, 3
3: 1, 2, 4
4: 3

$G.adj[u]$
↳ list of u's neighbors

Adj. matrix
- in a matrix, 1 = edge, 0 = no edge

	1	2	3	4
1	0	1	1	0
2	1	0	1	0
3	1	1	0	1
4	0	0	1	0

$G.A[u][v]$
↳ cell at position u, v

General run-time
↳ explore all edges

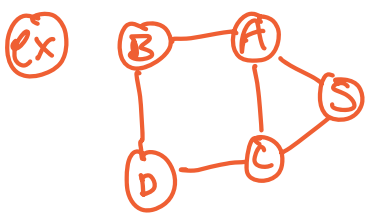
adj list $|V| + 2|E| = \Theta(V+E)$

adj matrix $|V|^2 = \Theta(V^2)$

2. Breadth-First Search

↳ "search" – find all reachable vertices from source vertex

Assume: undirected, unweighted, simple



start at S
- goal: find a path from s to every reachable vertex
- valid: path from s to all other vertices
- optimal: shortest path!

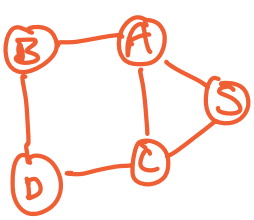
S to D
S, A, C, D
S, C, D
S, A, B, D

Algo Setup:
- white – haven't visited
- gray – in process
- black – done

vertex attributes, v:
- $v.color$
- $v.d$ – distance from s ← value of optimal
- $v.\pi$ – predecessor to v in path from s ← actual optimal solution

Preprocessing:

	S	A	B	C	D
color	g	w	w	w	w
d	0	∞	∞	∞	∞
π	nil	nil	nil	nil	nil



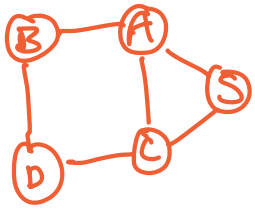
```
BFS(G, s)
 1  for each vertex u ∈ G.V − {s}
 2      u.color = WHITE
 3      u.d = ∞
 4      u.π = NIL
 5  s.color = GRAY
 6  s.d = 0
 7  s.π = NIL
 8  Q = ∅
 9  ENQUEUE(Q, s)
10  while Q ≠ ∅
11      u = DEQUEUE(Q)
12      for each v ∈ G.Adj[u]
13          if v.color == WHITE
14              v.color = GRAY
15              v.d = u.d + 1
16              v.π = u
17              ENQUEUE(Q, v)
18      u.color = BLACK
```

preprocessing

↳ at the end...

	S	A	B	C	D
color	b	b	b	b	b
d	0	1	2	1	2
π	nil	S	A	S	C

3. BFS Proof → result in table (vertex attributes)



- for every v, v.d = length of path from s to v
  - ↳ visit each vertex, start at source
    distance to that vertex is predecessor + 1
- for every v, v.π = predecessor in path
  - ↳ construct actual optimal soln

ex) D → C → S

D.π = C C.π = S

(S, C, D) length = 2

Q: S, A, C, B, D

What gets dequeued?

1. S, S.d = 0
2. A, A.d = 1
3. C, C.d = 1
4. B, B.d = 2
5. D, D.d = 2

- every vertex is enqueued once, dequeued once
  - ↳ only enqueue white vertices
- if v is dequeued after u
  v.d ≥ u.d
- when v is dequeued, we visit see its neighbors

BFS Proof: BFS gives an optimal solution

- valid: for every reachable vertex v from source s, valid path

- optimal: the path is a shortest one

Proof by induction!!

defs

$v.d$ = distance from s to v produced by BFS

goal: $v.d = \delta(s,v) \quad \forall v \in V$

$\delta(s,v)$ = length of shortest path

Induction on: $v.d$

- Base case $v.d = 0$ only for $s.d$, which is correct $\delta(s,s) = s.d = 0$ ☑

IH all vertices u such that $u.d \leq k$, $u.d = \delta(s,u)$

$s \rightsquigarrow u$ $\quad u.d \leq k$, $u.d = \delta(s,u)$

consider vertex w such that $w.d = k+1$

we must have a vertex v, an edge (v,w), and $v.d = k$

$s \rightsquigarrow v \to w$ $\quad v.d = k$, $w.d = k+1$

by IH $v.d = k = \delta(s,v)$

from here, show $s \rightsquigarrow w$ with $w.d = k+1$ is optimal

→ assume not!

assume: some path $s \rightsquigarrow w$ exists with length $\leq k$

w has predecessor v_0

$s \rightsquigarrow v_0 \to w$

v_0 would need to have $v_0.d \leq k-1$

But!! $v_0.d \leq k-1$, $v.d = k$

v_0 would have been dequeued first! and we would have discovered w before we got to v

↔ this is a contradiction!

So, $s \rightsquigarrow v \to w$ with $w.d = k+1$ is shortest path