Past: ext 1 more day

Past stuff

\[
\text{init. } S = (1, 2, 3, \ldots, n)
\]

\[
\text{DFS}(G, S)
\]

\[
S = (\ldots \text{dec. fin. time} \ldots) \ast
\]

\[
T = C^T
\]

\[
\text{DFS}(T, S)
\]

\[
S = (\ldots \text{dec. fin. time, 2nd call} \ldots)
\]
how to accomplish *?

One way:

split S into 2 sections

\[ S \quad \text{...........} \quad \text{...........} \]

\[ \text{list} \quad \text{stack} \]

initially:

\[ S \quad x \quad y \quad \ldots \quad \frac{1}{1} \quad \text{cursor} \quad \text{empty} \]

- front(S)
- deleteFront()
- iterate main load of DFS

- insertAfter() \Rightarrow Push
- at every finish event in Visit
Priority Queue ADT based on Heap.

**HeapMaximum (A)**

- **Pre:** `HeapSize(A) \geq 1`
- Time: $\Theta(1)$
  
  1. Return `AL1`

**HeapDeleteMax (A)**

- **Pre:** `HeapSize(A) \geq 1`
  
  1. `AL1 := A[heapSize[A]]`
  2. `heapSize[A] --`
  3. `Heapify(A, 1)`

**HeapExtractMax (A)**

- **Pre:** `HeapSize(A) \geq 1`
  
  1. `max := AL1`
  2. `HeapDeleteMax(A)`
  3. Return `max`

Time: $\Theta(\log n)$
HeapIncreaseKey \( (A, i, k) \)  
Pr.: 
\[ 1 \leq i \leq \text{heapSize}(A) \]

1. \( i \) \text{ if } k > A[i] 

2. \( A[i] = k \)

3. while \( i \geq 2 \) and \( A[\text{parent}(i)] < A[i] \)

4. \( A[i] \leftarrow A[\text{parent}(i)] \)

5. \( i = \text{parent}(i) \) \( \Theta(\log n) \)

HeapInsert \( (A, k) \)  
Pr.: \( \text{heapSize}(A) < \text{length}(A) \)

1. \( \text{heapSize}(A) + + \)

2. \( A[\text{heapSize}(A)] = -\infty \)

3. HeapIncreaseKey \( (A, \text{heapSize}(A), k) \) \( \Theta(\log n) \)
Exercise

Write: \text{Minimum}(1), \text{DeleteMin}()
\text{ExtractMin}(1), \text{DecreaseKey}() \text{ for a min P.Q.}

\text{Note:}

\text{General P.Q.} \quad \text{Our Picture}

\begin{align*}
\text{record} & \quad \text{key} \\
\text{Satellite data} & \quad \text{no satellite data}
\end{align*}
Exercise
write all algorithms in the general picture.

Exercise
implement min P.A. in
  * Java
  * C
SSSP in a weighted graph

(chap 24 in CLRS).

**Defn**
A weighted graph is a graph
\( G = (V, E) \) with a weight
function on edges
\[ w : E \rightarrow \mathbb{R} \]

**Defn** weight of a Path
\[ P : x = v_0, v_1, \ldots, v_k = y \]
\[ w(P) = \sum_{i=1}^{k} w(v_{i-1}, v_i) \]
**Defn** shortest path weight

\[ f(x,y) = \begin{cases} \inf \{ w(p) \mid p \text{ is an } x-y \text{ path} \} & \text{if } y \text{ reachable from } x \\ \infty & \text{otherwise} \end{cases} \]

**Defn** shortest \(x-y\) path \(P\)

A path \(P\) from \(x\) to \(y\)

\[ w(p) = f(x,y) \]

**SSSP Problem:**

Given \(G = (V,E), w : E \to \mathbb{R}\), and \(s \in V\), find \(f(s,x)\) for all \(x \in V\). For those \(x\) reachable from \(s\), determine a shortest \(s-x\) path.
we assume $G$ is a digraph.

Two algorithms

- Dijkstra (neg. + weights)
- Bellman-Ford (allows - weights)

Note: both actually find min-weight values

\[ w(x, y, z, x) = -5 \]
neither algorithm can work if \( s \) is a neg. weight cycle reachable from \( s \).

\[ \begin{aligned} \text{Vertex Attributes} \\
\text{PL}_x \downarrow : \text{parent of } \ x \\
\text{DL}_x \downarrow : \text{estimate of } \ \delta(s, x) \end{aligned} \]

Predecessor Subgraph

\[ \begin{aligned} V_p = \{ x \in V(G) \mid \text{PL}_x \neq \text{nil} \} \\
E_p = \{ (\text{PL}_x, x) \mid \text{PL}_x \neq \text{nil} \} \end{aligned} \]

To find shortest path see \text{PointPath}(G, s, x) \quad p. 602 \]
Subroutine

Initialize(G, s)

1.) for all x ∈ V(G)
2.) d[x] = ∞
3.) p[x] = nil
4.) d[s] = 0

Relax(x, y)  pre: y ∈ adj[L[x]]

1.) if d[y] > d[x] + w(x, y)
2.) d[y] = d[x] + w(x, y)
3.) p[y] = x
note:
- Relax changes at most the fields of $y$
- after Relax$(x,y)$, then

$$d(y) \leq d(x) + w(x,y)$$

must be true
Lemma 1

let \( x \in V(G) \). Suppose after Initialize\((G, s)\), some sequence of calls to Relax\((i, j)\) causes \( d[i] \) being finite. Then \( G \) contains an \( s \righ\) path of weight \( d[i] \).

Proof: Induction on \# of calls to Relax\((i, j)\).

starting at \( n=0 \) calls.

Exercise: Complete this.
Lemma 2

After Initialize(G, s), the inequality

\[ d(s, x) \leq d_L x \quad (\forall x \in V) \]

is true over any seq. of calls to relax.

Proof. (Contradiction)

Assume \( d_L x < d(s, x) \) for some \( x \), and some seq. of calls to Relax(\( x \), \( y \)).

Then \( d_L x \) is necessarily finite. By Lemma 1, G contains an s-x path
at weight \( \frac{\partial f(s, x)}{\partial s} \). This violates the definition of \( g(s, x) \). No such relaxation exists.