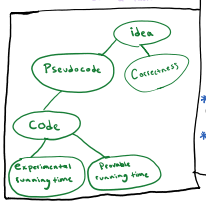
Notes 4

Thursday, September 9, 2021 11:03 AM

Algorithms

- What is an algorithm?
  - Step-by-step instructions to do a task



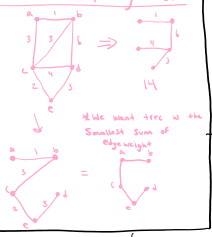
Steps of an Algorithm Design

1. Idea/Intuition
2. Pseudocode (use-resume level)
3. Code (details matter) — Iterative Process
4. Evaluate the experimental running time
5. * Prove the running time/optimal time
6. * Proof of correctness (works in every case scenario)

Technique: Greedy Algorithms

- Assume that the best solution to the problem instance overall is to choose the best local or short-term answer
- * think of navigating downtown... 1st step should be in direction of dt

Minimum Spanning Trees



* We want tree w the smallest sum of edge weights

Pseudocode for MSTs

```
While (the instance isn't solved)
{
  Select an edge according to
    some "locally optimal" criterion   // Selection Step
  if (adding edge does not create
      a cycle)                         // Feasibility check
    add it to the solution F
  if (F is a spanning tree of G)       // Solution check
    Win, output result & terminate
}
```

Proof of Correctness:

Let $G=(V,E)$ be an undirected weighted graph where $|V|=n$ and $|E|=m$.

A subset $F \subseteq E$ is called promising if we can add edges to F that results in a minimum spanning tree.

Proof (by induction on the promising nature of F):

b.c: 0 iterations of Prim's algorithm: $F = \emptyset$ is promising by definition

hypothesis: Suppose that F is promising after some # k $(k < n-1)$ iterations of Prim's alg.

Inductive Step: We want to show that adding an edge e that satisfies the local criteria yields a promising set.

In math: $F \cup \{e\}$ is promising.

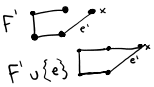
other words....

F is promising $\Rightarrow$ $F \subseteq F'$ and $G'=(V,F')$ is an MST

Case 1: $(e \in F')$

$F \cup \{e\} \subseteq F'$ → this is obviously true and therefore the correctness holds.

Case 2: $(e \notin F')$



$F' \cup \{e\}$ MUST contain a cycle

Prim's algorithm gives us....

weight(e) < weight(e')

$T = F' \cup \{e\} - \{e'\}$ * replace e' with e

has an overall lower weight than F'.

∴ F' is not an MST

∴ Prim did me right

CONTRA

Pseudocode for Prim's Algorithm

```
F = ∅
Y = {v_1}
While (not solved)
{
  O(m) Pick a vertex in V-Y that is nearest to Y   (other ^ n-1 times)
  O(1) add that vertex to Y
  O(1) add that corresponding edge to F
  *O(n) if (Y = V) *or check size
    Win (and quit);
}
```

→ Can it be better? Yes!!

$O(n)$

$\Rightarrow O(n)\left(O(m)+O(1)+O(1)+O(n)\right)$

$= O(n(m+1+n))$

$= O(n(m+n))$ $\quad m = n^2$

$O(n(n^2))$

$O(n^3)$

$|V| = n$ (ancy)

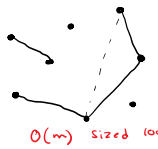
$|E| = m$ (eredith)

- The $O(m)$ step in Prim's is actually $O(n)$ using some simple code trick
- Comparing $Y=V$ can be done in $O(1)$ time using a simple coding trick

Why'd we screw up??

→ Because we did $O(1)$ analysis on the pseudocode

$O(n^2)$



$O(m)$ sized loop

$O(\log n)$ { Find(i) → tell me set i is in; Find(j)

$O(m \log n)$

Recall: $n + (n-1) + (n-2) + \ldots + 1 =$

$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = O(n^2)$

* When m is small, Kruskals better, for big m, Prim's is better

$m \log m = n^2$. Solve for m

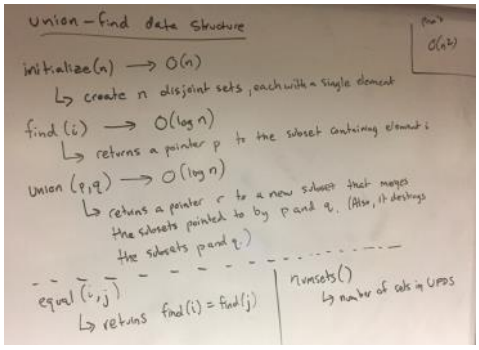
Let $m = \frac{n^2}{\log n}$

$$\frac{n^2}{\log n} \cdot \log\left(\frac{n^2}{\log n}\right) = \frac{n^2}{\log n}\left[\log(n^2) - \log(\log n)\right]$$

$$\leq \frac{n^2}{\log n} \cdot \log(n^2)$$

$$\leq \frac{n^2}{\log n} \cdot 2\log n$$

$$= 2n^2 \in O(n^2)$$



Kruskal's Alg (with 100% more UFDS)

```
F = ∅
initalize(n)
E.sort()
While (no solution)
{
  take smallest edge e = (i,j)
  if (find(j) != find(j))
  {
```

```
if (find(i) != find(v))
{
    F = F ∪ {e};
Union(
```