CS420 - Homework #1

Trevor Bramwell

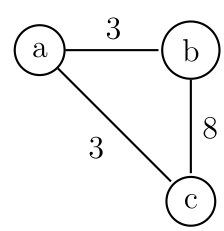
February 6, 2014

Problem 1

Throughout the lecture, we assumed that no two edges in the input graph have equal weights, which implies that the minimum spanning tree is unique. In fact, a weaker condition on the edge weights implies MST uniqueness.

(a)

Describe an edge-weighted graph that has a unique minimum spanning tree, even though two edges have equal weights.



(c)

Describe and analyze an algorithm to determine whether or not a graph has a unique minimum spanning tree.

Shimble's algorithm with the minor change that in the second loop:

Algorithm 1 Modified Shimble

...

for every edge $u \to v$ **do**
  if $u \to v$ is tense **then**
    return False
  end if
  return True
end for

Problem 2

Consider the following algorithm for finding the smallest element in an unsorted array:

Algorithm 2 RandomMin $A[1 \ldots n]$

$min \leftarrow \infty$
for $i \leftarrow 1$ to n in random order **do**
  if $A[i] < min$ **then**
    $min \leftarrow A[i]$ $(*)$
  end if
end for
return min

(a)

In the worst case, how many times does RandomMin execute line $*$?

$$O(n)$$

(b)

What is the probability that line $*$ is executed during the nth iteration of the for loop?

$$\frac{\binom{n}{1}}{n!} = \frac{1}{(n-1)!}$$

This states that there are $\binom{n}{1}$ ways to organize the first $n-1$ elements of RandomMin out of a total of $n!$ ways the array can be organized.

Problem 3

The randomized linear time algorithm of Karger, Klein and Tarjan for finding a minimum spanning tree performs two steps of the Boruvka algorithm and uses sampling steps in which each edge of the graph is chosen, independently, with probability 1/2.

The following solutions build off of the equation $\mathbb{E}[T(G)]$ presented by Uri Zwick the Lecture notes for "Analysis of Algorithms": Minimum Spanning Trees.

(a)

What would be the running time of the algorithm in terms of m, n, and k if it performs k steps of the Boruvka algorithm instead, where k is a positive integer?

When taking k steps of the Boruvka algorithm, the expected runtime changes to:

$$\mathbb{E}[T(G)] = a(\bar{m} + \bar{n}) + \mathbb{E}[T(G_1)] + (k-1)\mathbb{E}[T(G_2)]$$

This changes the upper bound to:

$$\begin{aligned}
\mathbb{E}[T(G)] &\le a(\bar{m} + \bar{n}) + 2a\left(\frac{\bar{m}}{2} + 2 \cdot \frac{\bar{n}}{4}\right) + (k-1)\left[2a\left(\frac{\bar{n}}{2} + 2 \cdot \frac{\bar{n}}{4}\right)\right] \\
&\le a(\bar{m} + \bar{n}) + 2a\left(\frac{\bar{m}}{2} + 2 \cdot \frac{\bar{n}}{4}\right) + 2a[(k-1)\bar{n}] \\
&= 2a(\bar{m} + k\bar{n})
\end{aligned}$$

(b)

What would be the expected running time of the algorithm in terms of m, n and p if the sampling probability were changed to p , where $0 \le p \le 1$?

The sampling probabilty only affects $\bar{m}_1$ in the original equation. Thus $\bar{m}_1 \leq \bar{m}p$

$$\begin{aligned}\mathbb{E}[T(G)] &\leq a(\bar{m}+\bar{n}) + 2a\left(\bar{m}p + \cdot 2\frac{\bar{n}}{4}\right) + 2a\left(\frac{\bar{n}}{2} + 2\cdot\frac{\bar{n}}{4}\right) \\ &\dots \\ &= 2a(\bar{m}(p+1) + 2\bar{n})\end{aligned}$$

Problem 4

(a)

Describe and analyze a modification of Shimbel's shortest-path algorithm that actually returns a negative cycle if any such cycle is reachable from s, or a shortest-path tree if there is no such cycle. The modified algorithm should still run in $O(mn)$ time.

Algorithm 3 ShimbelNegCycleOrTree

```
InitSSSP(s)
for all m do
  for all u → v do
    if u → v is tense then
      Relax(u → v)
    end if
  end for
end for
u_min ← s
for all u → v do
  if u → v is tense then
    return {recurse pred(u) until v}
  end if
  if dist(u) < dist(u_min) then
    u_min ← u
  end if
  return {recurse pred(u) until s}
end for
```

(b)

Describe and analyze a modification of Shimbel's shortest-path algorithm that computes the correct shortest path distances from s to every other vertex of the input graph, even if the graph contains negative cycles. Specifically, if any walk from s to v contains a negative cycle, your algorithm should end with $dist(v) = -\infty$ otherwise, $dist(v)$ should contain the length of the shortest path from s to v. The modified algorithm should still run in $O(mn)$ time.

Algorithm 4 ShimbelNegativeCycles

```
InitSSSP(s)
for all m do
    for all u → v do
        if u → v is tense then
            Relax(u → v)
        end if
    end for
end for
for all u → v do
    if u → v is tense or dist(u) = −∞ then
        dist(v) = −∞
    end if
end for
```