CMPS 101
Summer 2010
Homework Assignment 7 Solutions

1. (1 Point)
Let T be an almost complete binary tree on n nodes, and let $N(n,h)$ denote the number of nodes in T at height h, where $0 \le h \le \lfloor \lg(n) \rfloor$. Prove that

(*) $$N(n,h) = \left\lfloor \frac{n}{2^h} \right\rfloor - \left\lfloor \frac{n}{2^{h+1}} \right\rfloor .$$

Hint: first observe that the number of leaves in T is $n - \left\lfloor \frac{n}{2} \right\rfloor$.

Proof:
Let T be represented by an array A of length n, as is a Heap. Then the rightmost internal node in T is the parent of the rightmost leaf. (By rightmost we mean rightmost in the array A.) Thus the index of the rightmost internal node is $\text{parent}(n) = \lfloor n/2 \rfloor$.

$$\underbrace{A_1, A_2, \ldots\ldots, A_{\lfloor n/2 \rfloor}}_{\text{Internal Nodes}}, \quad \underbrace{A_{\lfloor n/2 \rfloor + 1}, \ldots\ldots, A_n}_{\text{Leaves}}$$

The number of leaves in T is therefore $n - \lfloor n/2 \rfloor$. But the leaves in T are precisely those nodes at height 0 in T, and hence $N(n,0) = n - \lfloor n/2 \rfloor = \lfloor n/2^0 \rfloor - \lfloor n/2^{0+1} \rfloor$. The formula (*) is thus seen to hold in the case $h = 0$.

Now let T' be the tree obtained by deleting all leaves in T (together with all incident edges.) Observe that all (remaining) nodes in T' have had their heights reduced by 1, since all paths from the root to a descendant leaf have been shortened by 1. In other words, if an internal node x had height h in T, then that same node x has height $h-1$ in T'. Observe also that T' is itself an almost complete binary tree having $n - (n - \lfloor n/2 \rfloor) = \lfloor n/2 \rfloor$ nodes. Therefore

$$N(n,h) = N\left(\lfloor n/2 \rfloor, h-1 \right)$$

From this we get

$$N(n,1) = N\left(\lfloor n/2 \rfloor, 0 \right) = \lfloor n/2 \rfloor - \left\lfloor \frac{\lfloor n/2 \rfloor}{2} \right\rfloor = \lfloor n/2^1 \rfloor - \lfloor n/2^2 \rfloor .$$

Formula (*) therefore holds in case $h = 1$ also. Similarly

$$N(n,2) = N\left(\lfloor n/2 \rfloor, 1 \right) = \left\lfloor \frac{\lfloor n/2 \rfloor}{2} \right\rfloor - \left\lfloor \frac{\lfloor n/2 \rfloor}{2^2} \right\rfloor = \lfloor n/2^2 \rfloor - \lfloor n/2^3 \rfloor ,$$

and (*) holds when $h = 2$. Continuing in this manner, we see that formula (*) holds for all *h* in the range $0 \le h \le \lfloor \lg(n) \rfloor$, as claimed. ///

Remark: This proof could have been phrased as an induction on *h*, starting at $h = 0$. This was not really necessary though, since there are only finitely many "dominos" to consider, namely those *h* in the range $0 \le h \le \lfloor \lg(n) \rfloor$.

2. (1 Point) p.132: 6.2-5
The code for Max-Heapify is quite efficient in terms of constant factors, except possibly for the recursive call in line 10, which might cause some compilers to produce inefficient code. Write an efficient Max-Heapify that uses an iterative control construct (a loop) instead of recursion.
Solution:
Max-Heapify(*A*, *i*)

```
1.  s ← heap-size[A]
2.  isHeap ← false
3.  while i ≤ parent(s) and not isHeap
4.        l ← left(i)
5.        r ← right(i)
6.        if  l ≤ s and A[l] > A[i]
7.              largest ← l
8.        else
9.              largest ← i
10.       if  r ≤ s and A[r] > A[largest]
11.             largest ← r
12.       if  largest ≠ i
13.             A[i] ↔ A[largest]
14.             i ← largest
15.       else
16.             isHeap ← true
```

3. (1 Points)
Let $x \in V(G)$ and suppose that after INITIALIZE-SINGLE-SOURCE(*G*, *s*) is executed, some sequence of calls to Relax( ) causes $d[x]$ to be set to a finite value. Then *G* contains an *s*-*x* path of weight $d[x]$. (Hint: Use induction on the length of the Relaxation sequence, and recall that this result was proved in class.)
Proof: Let *n* denote the length of the relaxation sequence. If $n = 0$, then the only *d*-value which is finite (after Initialization) is that of the source *s*. Indeed, *G* does contain an *s*-*s* path of weight $d[s] = 0$, namely the trivial path. The base case is therefore verified.

Let $n > 0$, and assume for any vertex *x*, that if $d[x]$ achieves a finite value during a sequence of fewer than *n* relaxations, then there exists an *s*-*x* path in *G* of weight $d[x]$. Now let $y \in V(G)$ and consider a sequence of *n* relaxations in which $d[y]$ becomes finite. Then an edge of the form (x, y) must have been relaxed during this sequence, and on that relaxation step, $d[y]$ was set to $d[x] + w(x, y)$. Since we suppose that this number is finite, $d[x]$ must have been finite before Relax(x, y) was executed. Thus $d[x]$ became finite during a sequence of fewer than *n* relaxations, and by our induction hypothesis, there must exist an *s*-*x* path in *G* of weight $d[x]$. That path, followed by the edge (x, y), constitutes an *s*-*y* path in *G* from of weight $d[x] + w(x, y) = d[y]$. ///

4. (1 Point) p.591: 24.1-3
Given a weighted, directed graph $G=(V,E)$ with no negative-weight cycles, let *m* be the maximum over all pairs of vertices $u,v\in V$ of the minimum number of edges in a shortest path from *u* to *v*. (Here, the shortest path is by weight, not the number of edges.) Suggest a simple change to the Bellman-Ford algorithm that allows it to terminate in $m+1$ passes.

Solution:
We reproduce Bellman-Ford here for reference:

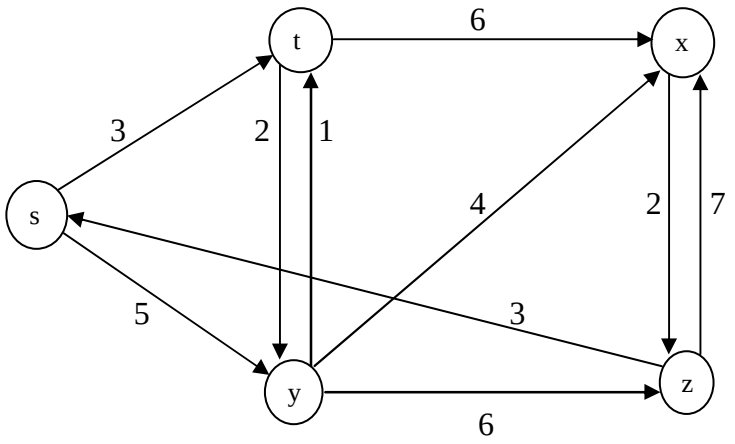
BELLMAN-FORD(*G*, *s*)
1. INITIALIZE-SINGLE-SOURCE(*G*, *s*)
2. for $i\leftarrow 1\,\text{to}\,|V|-1$
3. for each edge $(u,v)\in E$
4. RELAX(*u*, *v*)
5. for each edge $(u,v)\in E$
6. if $d[v]>d[u]+w(u,v)$
7. return false
8. return true

Note that we cannot simply alter the **for** statement on line 2 to say "for $i\leftarrow 1\,\text{to}\,m+1$", since the value of *m* is not known ahead of time. Instead we modify Bellman-Ford so that loop 2-4 terminates as soon as one complete pass over the edge set results in no *d*-values being changed. Obviously no *d*-values will be changed by performing any further passes, so if we accept the correctness of Bellman-Ford (Lemma 24.2 and Theorem 24.4), the *d* and π values must be correct at that point. It remains only to show that this rule causes loop 2-4 to terminate after $m+1$ passes. To prove this it is sufficient to show that the *d*-values are correct after exactly *m* passes. This follows from the path-relaxation property mentioned on p.587 (Lemma 24.15) which says:

> If $p=(v_0,v_1,\ldots,v_k)$ is a shortest path from $s=v_0$ to v_k, and the edges of *p* are relaxed in the order (v_0,v_1), (v_1,v_2), …., (v_{k-1},v_k), then $d[v_k]=\delta(s,v_k)$. This property holds regardless of any other relaxation steps that occur, even if they are intermixed with relaxations of the edges of *p*.

Each of the edges (v_0,v_1), (v_1,v_2), …., (v_{k-1},v_k) will be relaxed exactly once on each pass over the edge set, so *k* iterations of loop 2-4 suffice to correctly set the *d*-value of v_k. But by our definition of *m*, every vertex *v* (which is reachable from *s*) lies at the end of a shortest *s*-*v* path containing at most *m* edges, hence *m* iterations suffice to correctly set the *d*-values of all vertices in *G*. ///

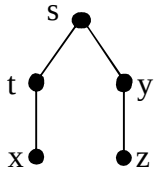
5. (1 Point) p.600: 24.3-1
Run Dijkstra's algorithm on the directed graph of Figure 24.2, first using vertex *s* as the source and then using vertex *z* as the source. In the style of Figure 24.6, show the *d* and π values and the vertices in set *S* after each iteration of the **while** loop.



Solution:
With *s* as source:

	d/π-values after *i*-th iteration of while					
Vertex	0	1	2	3	4	5
s	0/ nil	0/ nil	0/ nil	0/ nil	0/ nil	0/ nil
t	∞/ nil	3/ s	3/ s	3/ s	3/ s	3/ s
x	∞/ nil	∞/ nil	9/ t	9/ t	9/ t	9/ t
y	∞/ nil	5/ s	5/ s	5/ s	5/ s	5/ s
z	∞/ nil	∞/ nil	∞/ nil	11/ y	11/ y	11/ y
Set *S*	∅	{s}	{s, t}	{s, t, y}	{s, t, y, x}	{s, t, y, x, z}

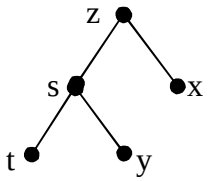
Predecessor subgraph:



With *z* as source:

	d/π-values after *i*-th iteration of while					
Vertex	0	1	2	3	4	5
s	∞/ nil	3/ z	3/ z	3/ z	3/ z	3/ z
t	∞/ nil	∞/ nil	6/ s	6/ s	6/ s	6/ s
x	∞/ nil	7/ z	7/ z	7/ z	7/ z	7/ z
y	∞/ nil	∞/ nil	8/ s	8/ s	8/ s	8/ s
z	0/ nil	0/ nil	0/ nil	0/ nil	0/ nil	0/ nil
Set *S*	∅	{z}	{z, s}	{z, s, t}	{z, s, t, x}	{z, s, t, x, y}

Predecessor subgraph:



6. (10 Points) p.600: 24.3-4

We are given a directed graph $G = (V, E)$ on which each edge $(u,v) \in E$ has an associated value $r(u,v)$, which is a real number in the range $0 \le r(u,v) \le 1$ that represents the reliability of a communication channel from vertex *u* to vertex *v*. We interpret $r(u,v)$ as the probability that the channel from *u* to *v* will not fail, and we assume that these probabilities are independent. Give an efficient algorithm to find the most reliable path between two given vertices.

Solution:

Let *p* be a directed path from *x* to *y* consisting of the vertices: $x = v_0, v_1, v_2, \ldots, v_k = y$. Since the probabilities associated with each edge are independent, the probability that no edge along *p* fails is given by $r(p) = \prod_{i=1}^{k} r(v_{i-1}, v_i) = r(v_0, v_1) \cdot r(v_1, v_2) \cdots r(v_{k-1}, v_k)$. It is possible to modify Dijkstra's algorithm so that it will determine a path *p* which maximizes the quantity $r(p)$. This is the most reliable path from *x* to *y*, which we seek. However, it is easier to use Dijkstra's algorithm directly by carefully defining an appropriate weight function on edges. Given $(u,v) \in E$ we define $w(u,v) = -\log(r(u,v))$, where the log function can have any base greater than 1. Since $0 \le r(u,v) \le 1$ we have $-\infty \le \log(r(u,v)) \le 0$, and hence $0 \le w(u,v) \le \infty$. Thus the edge weights are non-negative, and possibly infinite. Therefore running Dijkstra's algorithm on the source *x* will determine an *x*-*y* path which minimizes the quantity

$$\begin{aligned} w(p) &= \sum_{i=1}^{k} w(v_{i-1}, v_i) \\ &= \sum_{i=1}^{k} -\log(r(v_{i-1}, v_i)) \\ &= -\sum_{i=1}^{k} \log(r(v_{i-1}, v_i)) \\ &= -\log\left(\prod_{i=1}^{k} r(v_{i-1}, v_i) \right) \\ &= -\log(r(p)). \end{aligned}$$

But then *p* must maximize the quantity $\log(r(p))$, and since log is an increasing function, the path *p* also maximizes $r(p)$ as required. The following algorithm determines the most reliable directed *x*-*y* path in *G*.

Max-Reliable(*G*, *x*, *y*, *r*)

1. for each $(u,v) \in E(G)$
2. $\quad w(u,v) \leftarrow -\log(r(u,v))$
3. Dijkstra(*G*, *w*, *x*)
4. return the minimum weight *x*-*y* path determined in step 3

///

7. (1 Point) p.613: 24.5-4
Let $G = (V, E)$ be a weighted, directed graph with source vertex s and let G be initialized by INITIALIZE-SINGLE-SOURCE(G, s). Prove that if a sequence of relaxation steps sets $\pi[s]$ to a non-NIL value, then G contains a negative-weight cycle.

Proof:

RELAX(u, v)
1. if $d[v] > d[u] + w(u,v)$
2. $\quad d[v] \leftarrow d[u] + w(u,v)$
3. $\quad \pi[v] \leftarrow u$

Examination of the above pseudocode for RELAX(u,v) shows that whenever the algorithm sets the predecessor $\pi[v]$ for some vertex v, it also reduces that vertex's d-value $d[v]$. Thus if $\pi[s]$ is at some point set to a non-NIL value, then $d[s]$ is reduced from its initial value of 0 to a negative number. But $d[s]$ is necessarily the weight of some existing path from s to s, by problem 1 of this assignment. This path is the required negative weight cycle in G. ///