FINAL EXAM SOLUTIONS
CSCI 163/COEN 179: THEORY OF ALGORITHMS
SPRING 2009

1. (5 points) Given a sorted array `A[1..n]` of distinct integers, write an $O(\log n)$ algorithm to determine whether `A[i] == i` for some `i`.

```
bool Q1(unsigned A[1..n])
{
   lower = 1;
   upper = n;
   while (lower <= upper)
   {
        mid = (lower + upper) / 2;
        if (A[mid] == mid)
             return true;
        if (A[mid] > mid)
             upper = mid - 1;
        else
             lower = mid + 1;
   }
   return false;
}
```

2. (5 points) Write an $O(|V| + |E|)$ algorithm `bool is_single_cycle(V, E)` to determine whether the input graph $G = (V, E)$ consists of a single simple cycle.

```
bool Q2(V, E)
{
     for (v = 0; v < |V|; ++v)
            if (neighbors(v).size() != 2)
                return false;
     v = 0;
     return connected(V, E);
}
```

3. (5 points) Write an algorithm `int Min(int H[1..n])` to return the smallest element in a max-heap `H` of size `n`. What is the asymptotic running time of your algorithm ?

```
int Q3(int H[1..n])
{
     answer = H[n/2 + 1];
     for (i = n/2 + 2; i <= n; ++i)
          if (H[i] < answer)
               answer = H[i];
     return answer;
}
```

The algorithm runs in time $\Theta(n)$.

4. (5 points) Write an $O(\log n)$ algorithm `unsigned powerof3(unsigned n)` to compute 3^n.

```
unsigned powerof3(unsigned n)
{
    if (n == 0)
       return 1;
    if (n % 2 == 0)
    {
         t = powerof3(n/2);
         return t*t;
    }
    else
    {
         t = powerof3(n/2);
         return 3*t*t;
    }
}
```

Each recursive call of `powerof3` reduces the argument by a factor of $\frac{1}{2}$ and hence the recursion depth is at most $O(\log_2 n)$.

5. (5 points) Write an $O(\log a + \log b)$ algorithm `unsigned lcm(unsigned a, unsigned b)` to compute the least common multiple of two positive integers `a` and `b`.

```
unsigned lcm(unsigned a, unsigned b)
{
    return (a*b)/gcd(a, b);
}
```

6. (5 points) Give an $O(n^2)$ algorithm to find the transitive closure of an undirected graph.

```
void tc(V, E)
{
      n = |V|;
      m = |E|;

      int A[n][n];

      for (i = 0; i < n; ++i)
          for (j = 0; j < n; ++j)
              A[i][j] = 0;

      bool marked[n] = {false};

      for (int i = 0; i < n; ++i)
          if (!marked[i])
          {
               S = bfs(i);   // S is the set of vertices reachable from i
               for each v in S
                  A[i][v] = A[v][i] = 1;
          }
}
```

7. (5 points) Write an algorithm to find the *maximum-weight* spanning tree of an input graph $G = (V, E, W)$.

```
maximum-weight-spanning-tree(V, E, W)
{
     for each edge e in E
          w(e) = -w
     return minimum-weight-spanning-tree(V, E, -W);
}
```

8. (5 points) Write an $O(mn)$ algorithm to determine whether it is possible to make change for a value n using denominations $d[1..m]$ *so that each denomination is used at most once.*

```
int coin-change-without-repetition(n, d[1..m])
{
     for (i = 0; i <= n; ++i)
            A[i][0] = 0;
     for (j = 0; j <= m; ++j)
            A[0][j] = 0;

     for (i = 1; i <= n; ++i)
            for (j = 1; j <= m; ++j)
                A[i][j] = max(A[i][j-1], 1 + A[i-d[j]][j-1]);
     return A[n][m];
}
```

The running time is clearly $\Theta(nm)$;

9. (5 points) Find a Huffman code for the alphabet {A, C, G, T} whose probabilities are 0.35, 0.2, 0.05 and 0.4 respectively.

```
A: 11
C: 101
G: 100
T: 0
up to isomorphism.
```

10. (5 points) Show that the LONGEST PATH problem is in NP:

- INPUT: an undirected graph G and a positive integer L;
- OUTPUT: yes if and only if G has a simple path of length L.

```
Longest-Path-Verify(V, E, L, P)
{
     if (P.size() != L + 1)
        return false;

     for (i = 0; i < P.size() - 1; ++i)
         if (P[i], P[i+1]) is not an edge
              return false;

     sort(P);
     for (i = 0; i < P.size()-1; ++i)
         if (P[i] == P[i+1])
            return false;

     return true;
}
```

There are at most $n = |V|$ vertices in P and hence the running time of the above algorithm is $O(n+nm) = O(nm)$ which is polynomial in the input size. Hence LONGEST-PATH is in NP.