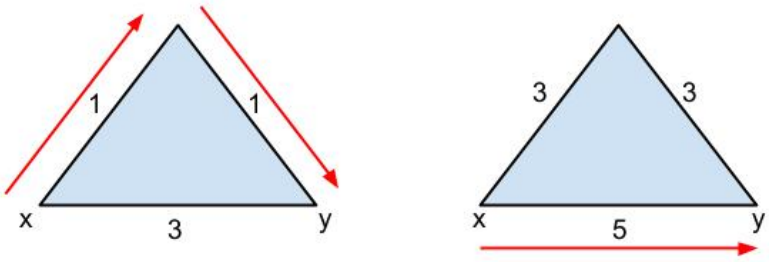
CSE 373 Analysis of Algorithms April 29, 2014

Solutions to Midterm 2

PROBLEM 1

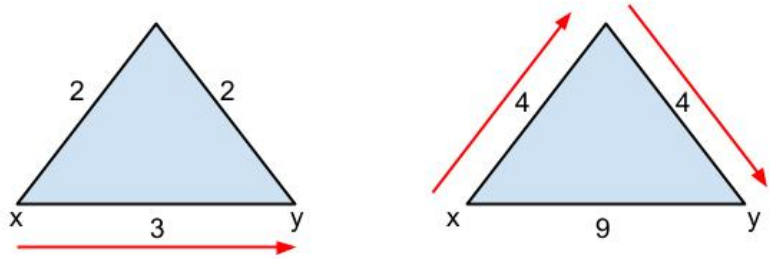
(a) On addition of a constant $c > 0$ to each edge, the minimum spanning tree remains the same. This is because the ordering of edges would still remain the same and hence Kruskal's algorithm will procude the same minimum spanning tree.

For the shortest path from x to y on the other hand, a path with fewer edges may become preferable. See counterexample below.



(b) Multiplication of a constant $c > 0$ to each edge does not change the minimum spanning tree or the shortest path. The weight of each just gets multiplied by c.

(c) Squaring the edges does not change the minimum spanning tree for the same reason as (a). But the shortest path may change. See counterexample below.



PROBLEM 2

This problem just asks you to verify whether the friendship graph is bipartite. This can be done using BFS/DFS by coloring the parents and their children alternatively using two colors in $O(n+m)$ time.

PROBLEM 3

This problem just asks you to verify if the graph can be paritioned into more than one connected component. You can do this using BFS/DFS in $O(n+m)$ time.

PROBLEM 4

Use Dynamic Programming with the following recurrence:

$D[i][j] = D[i][j-1] + D[i-d_j][j]$

Where $D[i][j]$ is the number of ways to make a change for i using at most j first denominations.

PROBLEM 5

Use backtracking algorithm from Skiena's Algoritms Design Manual. Array a will hold a sequence of used denominations in lexicographical order (to avoid duplicate solutions). We need to modify the following functions:

```
is_a_solution(sum,n) {
  return sum==n;
}
process_solution(a,k) {
    for (int i=0; i<k; i++) {
      print a[i]+‘‘,’’;
    }
}

construct_candidates(int[] a, int i, int[] d, int[] c,int *ncandidates) {
      for (int j=i; j<d.length; j++) {
            c[ncandidates++] = j;
      }
}

backtrack(int a[], int k, int n, int[] d, int i, int sum)
{
```

```
  int c[MAXCANDIDATES]; /* candidates for next position */
  int ncandidates; /* next position candidate count */
  if (is_a_solution(sum, n)) {
      process_solution(a,k);
  } else if(sum>n) {
      return;
  } else {
      k = k+1;
      construct_candidates(a,i,d,c,&ncandidates);
      for (int j=0; j<ncandidates; j++) {
            a[k] = d[c[j]];
            backtrack(a,k,n,d,c[j],sum+d[c[j]]);
      }
   }
}
```