CPSC 320 Midterm 2

July 13, 2007

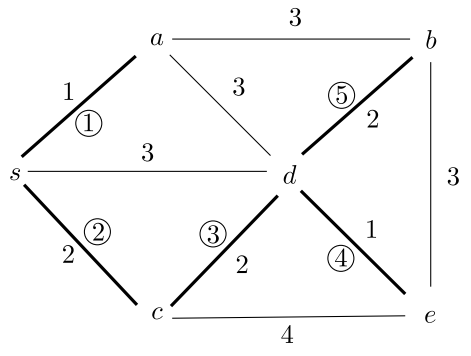
Name: _________________________ Student ID: ______________________

Signature: ______________________

- You have 1 hour to write the 5 questions on this examination. A total of 20 marks are available.
- *Justify all of your answers.*
- Keep your answers short. If you run out of space for a question, you have written too much.
- The number in the square brackets next to the question number is the # of marks allocated for that question. Use these to help determine how much time you should spend on each question.
- Use the back of the pages for your rough work.
- *Good luck!*

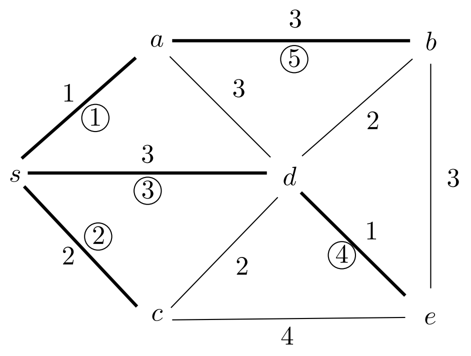
Question	Marks	Score
1	2	
2	2	
3	7	
4	4	
5	5	
Total	20	

1. [2] Run Prim's algorithm on the weighted, undirected graph below. The seed vertex is s. Thicken the edges that are selected for the minimum spanning tree. Number the edges as they are added to the tree in order from 1 to 5. If there is more than one correct answer, any correct answer will suffice.



2. [2] Run Dijkstra's algorithm on the weighted, undirected graph below. The source vertex is s. Thicken the edges that are selected for the shortest path tree. Number the edges as they are added to the tree in order from 1 to 5. If there is more than one correct answer, any correct answer will suffice.

The order of 4 and 5 may be reversed.



3. [7] You are given a pile of n coins of varying values $c_1, c_2, \ldots, c_n$, where each coin has a value between 1 and C cents. Is it possible to divide the coins between two people so that each person gets the **same** amount of money? The following algorithm solves the problem recursively. Let $S(k) = \sum_{i=1}^{k} c_i$. Then the call `Split(Coins,`$k$`,`$S(k)/2$`,`$S(k)/2$`)` returns `True` if the first k coins can be split into two equal piles.

```
// can the first $k$ coins can be split into piles of Person1 and Person2
Algorithm Split(int Coins[], int $k$, int Person1, int Person2)
   if (Person1$< 0$) or (Person2$< 0$) then
      return False
   if ($k = 0$) then
      return (Person1 $= 0$) and (Person2 $= 0$)
   else return
      Split(Coins,$k-1$,Person1-Coins[k],Person2) or // $k$th coin to
      Split(Coins,$k-1$,Person1,Person2-Coins[k])    // person 1 or 2
```

(a) [3] Use memoization to make this algorithm efficient. Describe what each cell of the table (cache) represents, and give a modified recursive algorithm that uses the table (fills it in).

Let $P[p, q, k]$ indicate if the first k coins can be split so that person 1 gets p coins and person 2 gets q coins. Set it to -1 if it is not yet computed, 0 if it is impossible, and 1 if it is possible. The dimensions of P are $(nC + 1) \times (nC + 1) \times n$. Initialize every element of P to -1. Then call the recursive procedure below:

```
Algorithm Split(int Coins[], int $k$, int Person1, int Person2)
   if (Person1$< 0$) or (Person2$< 0$) then
      return False
   if ($k = 0$) then
      return (Person1 $= 0$) and (Person2 $= 0$)
   if ($P$[Person1, Person2, $k$] $\neq -1$) then
      return $P$[Person1, Person2, $k$]
   else
      $P$[Person1, Person2, $k$] $\leftarrow$
         Split(Coins,$k-1$,Person1-Coins[k],Person2) or
         Split(Coins,$k-1$,Person1,Person2-Coins[k])
      return $P$[Person1, Person2, $k$]
```

(b) [4] Eliminate a dimension of the table using the fact that $p + q = S(k)$, where p and q are any split of the first k coins. That is, if you know the total of all the coins (right hand side) and the total of one part of the split (p), then you know the total of the other part of the split (q). Write your solution iteratively (i.e. using `for` loops instead of recursion).

Let $Q[i, k] = P[i, S(k) - i, k]$ = if it is possible to split the first k coins into piles of sum i and $S(k) - i$. The $Q[i, k]$ can be defined recursively.

$$Q[i,k] = \begin{cases} \text{True} & \text{if } i = 0 \text{ and } k = 0 \\ \text{False} & \text{if } i \neq 0 \text{ and } k = 0 \\ Q[i, k-1] & \text{if } i < c_k \\ Q[i - c_k, k-1] \vee Q[i, k-1] & \text{otherwise} \end{cases}$$

The recursion always refers to smaller values of k, so fill the table from small values of k to large values.

4. [4] Professor Midas drives an automobile from Newark to Reno along Interstate 80. His car's gas tank, when full, holds enough gas to travel n miles, and his map gives the distances between gas stations on his route. The professor wishes to make as few gas stops as possible along the way.

   (a) [2] Give an efficient method by which Professor Midas can determine at which gas stations he should stop.

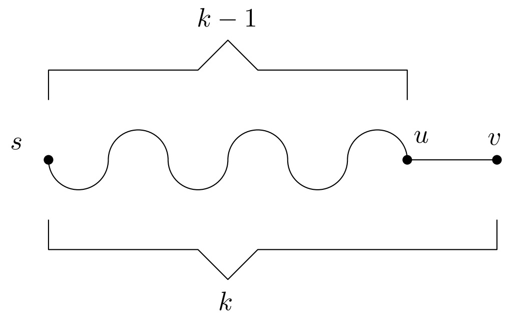
   He should always wait as long as possible to fill his tank, while ensuring that he never runs out of gas. Specifically, he will fill up at a gas station if he does not have enough gas left in his tank to make it to the next one.

   (b) [2] Argue that some filling schedule makes your first greedy choice.

   Let S be an optimal filling schedule and let s be the first stop in this schedule and u be the stop immediately after s. If s is not our greedy choice t, we can substitute t for s: by our greedy criteria, t comes after s; therefore the distance from t to u is less than the distance from s to u; hence the schedule resulting from the swap is feasible.

5. [5] Suppose you are given a diagram of a telephone network, which is a weighted, undirected graph G whose vertices represent switching centers, and whose edges represent communication lines between two centres. The edges are marked by their bandwidth. The bandwidth of a path is the bandwidth of its **lowest** bandwidth edge. Modify the **Bellman-Ford** algorithm so that given a diagram and a switching center s, it will compute the maximum bandwidth of a path from s to every other vertex. Breifly explain why it works.

Let $B[v, k]$ be the maximum bandwidth of paths from s to v using at most k edges. Consider a maximum bandwidth path from s to v using exactly k edges.



Then we can substitute any maximum bandwidth path from s to u using $k-1$ edges. The result is a maximum bandwidth path from s to v using k edges. So we try all u.

$$B[v,k] = \begin{cases} \infty & \text{if } k = 0 \text{ and } v = s \\ 0 & \text{if } k = 0 \text{ and } v \neq s \\ \max \left\{ \begin{array}{c} B[v, k-1], \\ \max\{\min\{B[u,k-1], w(\{u,v\})\} : \{u,v\} \in E\} \end{array} \right\} & \text{otherwise} \end{cases}$$