CS3230
Tutorial 8

1. Prove or disprove:

   (a) Suppose the graph contains only one edge of minimal weight. Then show that every minimal spanning tree must contain that edge.

   Ans. Suppose some MST T does not contain the minimal weight edge e. Then consider the graph $T \cup \{e\}$. Then, this graph contains a cycle. Delete an edge from the cycle different from e. Then the resulting graph is a tree with weight less than T. A contradiction.

   (b) Suppose a graph may contain several edges of minimal weight. Then does the minimal spanning tree contain all the edges of minimal weight?

   Ans: No. Consider the graph:

   $V = \{1, 2, 3, 4\}$

   $wt(1, 2) = 1$, $wt(1, 3) = 1$, $wt(2, 3) = 1$, $wt(1, 4) = 2$

   Then, the MST contains two of $(1, 2), (2, 3), (1, 3)$

   AND

   $(1, 4)$.

2. For $n > d \geq 1$, show that $\sum_{i=d}^{n-1} \binom{i}{d} = \binom{n}{d+1}$.

   Answer: Choosing $d + 1$ elements from n different objects is same as:

   Choosing the first object (say as j-th object), and then choosing the d objects from the remaining $n - j$ objects.

   Thus, $\binom{n}{d+1} = \sum_{j=1}^{n} \binom{n-j}{d}$

   $= \sum_{j=1}^{n-d} \binom{n-j}{d}$ (as $\binom{m}{d}$, for $m < d$ is 0).

   $= \sum_{n-j=d}^{n-j=n-1} \binom{n-j}{d}$

   $= \sum_{i=d}^{i=n-1} \binom{i}{d}$

3. Consider the sequence of following operations in a binomial heap starting with empty heap:

   insert 0, insert 5, insert 3, insert 6, insert 2, insert 10, insert 7, insert 8, insert 9, delete min.

   Show how the binomial heap would look after the above operations.

   Answer: The final tree would have:

   Root: 2

   Children of 2: 10, 7 and 3

   Child of 7: 8

   Children of 3: 6 and 5

   Child of 5: 9

4. Consider the sequence of following operations in a Fibonacci heap starting with empty heap:

   insert 0, insert 1, insert 2, delete min.

   insert -1, insert -2, insert -3, delete min

   insert -4, insert -5, insert -6, delete min

   insert -7, insert -8, insert -9, delete min

   Show how the Fibonacci heap would look after the above operations

   Answer: The final tree would have:

   Root: -8

   Children of -8 : -7, -5, -2

   Child of -5 : -4,

   Children of -2: -1, 1

   Child of 1: 2

5. Suppose we use binary representation to keep a number in memory. That is, number n is represented using bits $A[k]A[k-1], \ldots A[1]$, where $A[1]$ is the least significant bit, and k is at most $1 + \log n$ (you may assume $A[k+1], A[k+2], \ldots$ are all 0).

   To increment the number, we do the following process:

Done=False

$i = 1$

While not done

    If $A[i] = 0$, then set $A[i] = 1$ and done=true

    Else set $A[i] = 0$

    EndIf

    Set $i = i + 1$.

End while

Note that if the value of the number is n, then it may take $O(\log n)$ steps to do the increment.

Show that amortized cost of increments is constant.

Answer: Consider the following way of keeping potential: If the bit is 0, then potential is 0; if the bit is 1, then the potential is 1.

Initially, all bits are 0 and thus total potential is 0.

When we add 1 to existing number, we flip some bits (say k) from 1 to 0 (work done is $O(k)$, and potenital is decreased by k), and flip one bit from 0 to 1 (cost is $O(1)$, and potential is increased by 1). Thus, amortized cost of increment is constant.

6. In previous question, suppose we do not start with value 0, but some arbitrary value $m \leq n$. Then, show that amortized cost of n increments is still constant, where one starts with some potential which is atmost $O(\log n)$.

   Answer: Anlyzing as in previous question, the initial potential is at most $\log n$, and the amortized cost is constant for the remaining operations. This would make the total cost at most $2n + \log n$