CS 344 Assignment 3

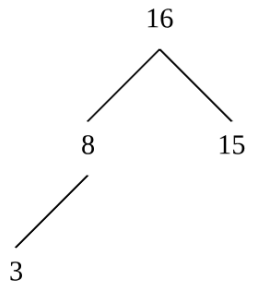
November 10, 2018

Name : Yang Bao
NetID : yb184
Section : 02
The classmate I discussed with : Zitian Qin

Problem 1 :

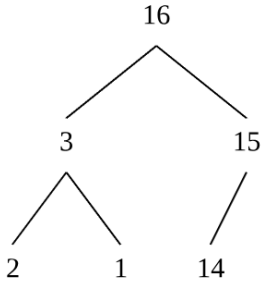
1. Part 1 :
   True. If the minimum element of a max-heap is not a leaf, then it must be a parent of its children. Since according to the property of max-heap, the child must be smaller than its parent, the child of that minimum element must be smaller than itself, so it is not the minimum element. So the minimum element of a max-heap must be a leaf.

2. Part 2 :
   False.



   8 is the second smallest element of that max-heap but not a leaf.

3. Part 3 :
   False.

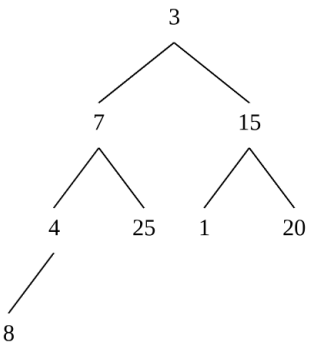


   14 is the third largest element of that max-heap but not a child of the root.

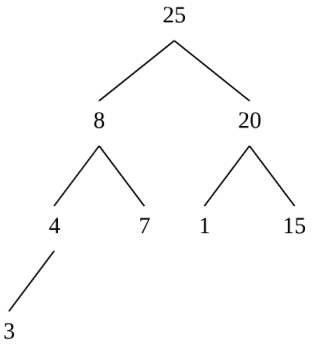
4. Part 4 :
   True. The sum of the heights of all nodes in an n-node heap is :
   $\sum_{h=1}^{log(n)} h \cdot \frac{n}{2^{h-1}} = n(1 + \frac{2}{2} + \frac{3}{4} + \cdots) = n \cdot \Theta(1) = \Theta(n)$

Problem 2 :

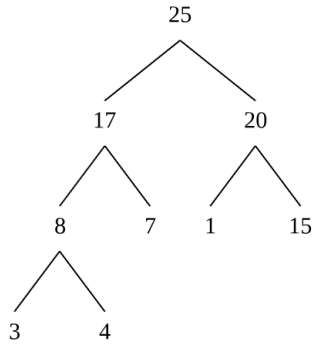
1. Part 1 :



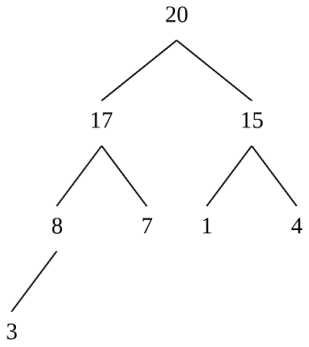
2. Part 2 :



3. Part 3 :



4. Part 4 :

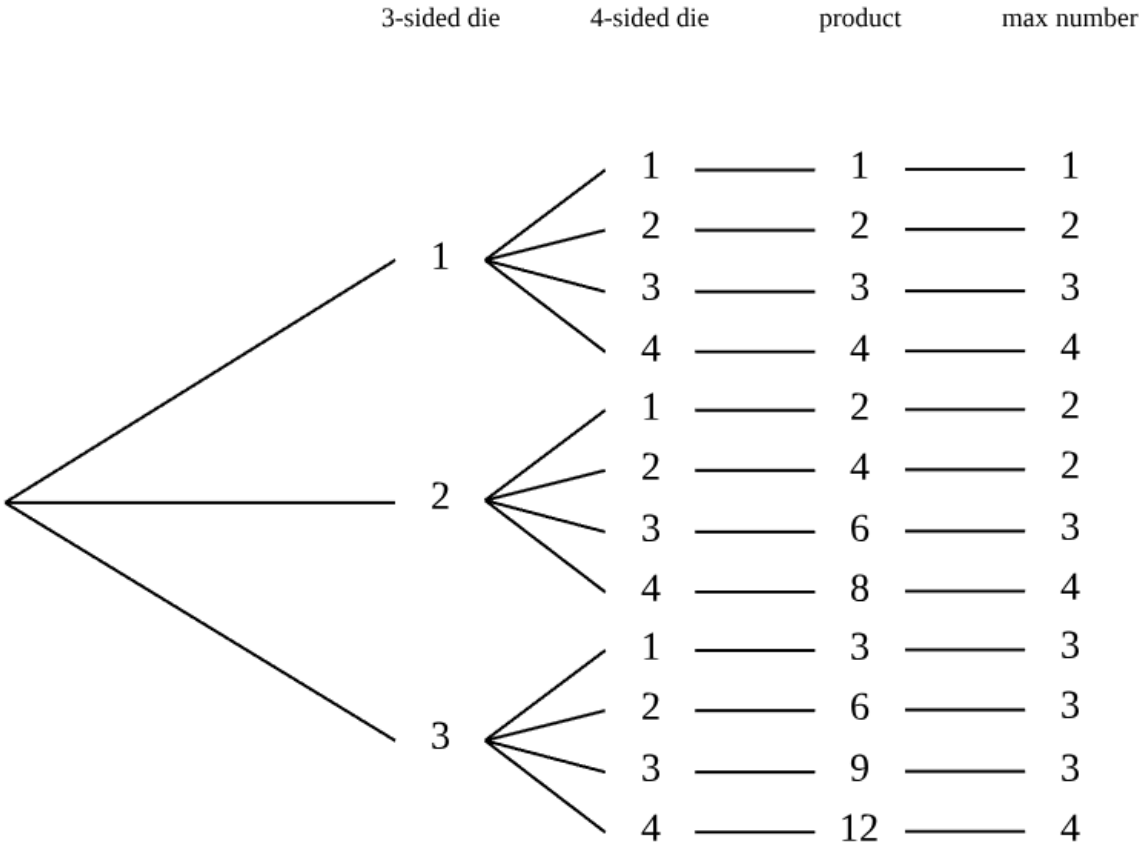


Problem 3 :
The pseudocode is :

```
1. build a min-heap of all n sorted arrays with A[0] as the keys in heap \\O(n)
2. for i = 1 to n
     output Heap.find-min()          \\O(1)
     Heap.delete-min()               \\(log n)
     Heap.add( the next element in that array)        \\O(log n)
```

I don't know if my pseudocode expresses what I truly mean so I want to explain more here. Since the arrays $A_1, A_2, ...A_n$ are all sorted array, $A_1[0], A_2[0], ...A_n[0]$ will be the smallest elements in each array. Building a min-heap of these n elements will take O(n) time. By using min-heap we can find the smallest in O(1) time. And after we find the smallest element, (assume that element is $A_2[0]$) we delete that element from the heap and add the next element in that array (which in this case is $A_2[1]$) into the heap and begin next loop. While the loop runs for n time, we will find n smallest elements in those n arrays.
The running time of the pseudocode is
$O(n) + n \cdot (O(1) + O(logn) + O(logn)) = O(n) + n \cdot O(logn) = O(nlogn)$ **Problem 4 :**

1. Part 1 :



Expectation of product $= \frac{1}{12} \cdot (1+2+3+4+2+4+6+8+3+6+9+12) = 5$
Expectation of max number $= \frac{1}{12} \cdot (1+2+3+4+2+2+3+4+3+3+3+4) = \frac{17}{6}$

2. Part 2 :
   First, since we have an array of n distinct integers, we will have $\binom{n}{2}$ pairs of (i, j).
   Let E_i be the indicator variable that the $i-th$ pair need to be swapped.
   Then the expect number of swapped pair $= \sum_{i=1}^{\binom{n}{2}} P_i$
   The probability of swap can be considered this way :
   Let set A be the set of all pairs of (i, j) that $i < j$, and set B be the set of all pairs of (i, j) that $i > j$.
   Since the integers are distinct, there won't be the situation that $i = j$, so $|A| + |B|$ will cover all the possible events.
   So the probability of swap will be $\frac{|A|}{|A|+|B|}$.
   Moreover, we can build a bijection between A and B, since for every pair in A, if we swap i and j then that will become a pair in B, so $|A| = |B|$.
   So the probability of swap will be $\frac{1}{2}$
   So the expect number of swapped pair $= \frac{1}{2} \times \binom{n}{2} = \frac{n(n-1)}{4}$

Problem 5 :

1. Part 1 :
   Let X_i be the indicator variable that the $i-th$ 3 number meets the pattern $3, 6, 6$.
   Since there 100 numbers so i will be $1 \leq i \leq 98$.
   So the expect number of times will be $\sum_{i=1}^{98} E[X_i]$.
   $E[X_i] = 1 \cdot P(X_i) = 1 \cdot \frac{1}{6} \cdot \frac{1}{6} \cdot \frac{1}{6} = \frac{1}{216}$ So the expect number of times will be $\frac{98}{216} = \frac{49}{108}$

2. Part 2 :
   Let X_i be the indicator variable that 3 people have a 3-way birthday.
   Let n be the number of people in a room. So there can form $\binom{n}{3}$ pair of 3 peoples.
   Since there 365 days in a year, $P(X_i) = \frac{1}{365^3}$.
   Then the expected number of 3-way-birthdays will be $\sum_{i=1}^{\binom{n}{3}} \frac{1}{365^3} = \frac{n(n-1)(n-2)}{3 \times 2 \times 365^3}$
   Now we need the expected be at least 2, so :
   $\frac{n(n-1)(n-2)}{3 \times 2 \times 365^3} \geq 2$
   $n(n-1)(n-2) \geq 12 \times 365^3$
   When $n = 836$, $n(n-1)(n-2) < 12 \times 365^3$
   When $n = 837$, $n(n-1)(n-2) > 12 \times 365^3$
   So we need at least 837 people.

Problem 6 - extra credit :
I think the expected number of objects I will look at before I have seen every one of the n objects E can be divided into :
$E = E_1 + E_2 + \cdots + E_n$ where E_i stands for the number of objects I looked to find the $i-th$ new object.
Obviously that $E_1 = 1$ because no matter which object we looked first, it will be a new object.
Let P_i be the probability that we look at an object and find it's new since we have found (i-1) objects.
And obviously $P_i = \frac{n-i}{n}$
According to the key fact from the lecture, Since $P_i = \frac{n-i}{n}$, $E_i = \frac{n}{n-i}$ So we can rewrite E as $E = 1 + \sum_{i=2}^{n} \frac{n}{n-i} = \sum_{i=1}^{n} \frac{n}{n-i} = \sum_{i=1}^{n} \frac{n}{n-i}$
$\sum_{i=1}^{n} \frac{n}{n-i} = n \cdot (1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n})$
According to the hint, we have that $1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} = O(logn)$
So we have $\sum_{i=1}^{n} \frac{1}{P_i} = n \cdot O(logn) = O(nlogn)$
So we have $E = O(nlogn)$