

## Part A. Probabilistic Analysis and Randomized Algorithms

### 1.Hat check problem

Let's define our indicator random variable as  $X_i = I \{ \text{event of } i\text{th customer gets his/her hat back} \}$  where  $1 \leq i \leq n$ .

Our goal is to find  $E[X]$ .

There exist  $n$  different hat so the probability of the right (own) hat is  $\frac{1}{n}$ . Mathematically,

$$\Pr \{ \text{event of customer gets his/her hat back} \} = \frac{1}{n}$$

From the Lemma 5.1 in lecture slides  $E[X_i] = \Pr \{ \text{event of customer gets his/her hat back} \} = \frac{1}{n}$

Also we have  $n$  customer and our indicator variable represents the status of one customer so  $X = X_1 + X_2 + \dots + X_n$

$$X = X_1 + X_2 + \dots + X_n = \sum_{i=1}^n X_i$$

Finally

$$E[X] = E \left[ \sum_{i=1}^n X_i \right] = \sum_{i=1}^n E[X_i] = \sum_{i=1}^n \Pr \{ \cdot \} = \sum_{i=1}^n \frac{1}{n}$$

So we can find  $E[X] = 1$

### 2.Alice problem

This time we have two different indicator random variable:

$X$  in order to represent the time and  $Y$  in order to represent the chosen door  $\{D1, D2, D3\}$

Our goal is to find  $E[X]$  again.

Here  $X$  is dependent with chosen door( $Y$ ) ( $X$  is dependent occurrence of  $Y$ ). So we can redefine our goal as  $E[X] = E[E[X|Y]]$  and from the definition of the expected value from the slides

$$E[X] = E[E[X|Y]] = E[X|Y=D1] \cdot \Pr\{D1\} + E[X|Y=D2] \cdot \Pr\{D2\} + E[X|Y=D3] \cdot \Pr\{D3\}$$

$$E[X|Y=D1] = 2 \text{ hour}$$

$$E[X|Y=D1] = 3 \text{ hour to comeback} + E[X]$$

$$E[X|Y=D1] = 5 \text{ hour to comeback} + E[X]$$

$$E[X] = E[E[X|Y]] = 2 \cdot \frac{1}{3} + (3 + E[X]) \cdot \frac{1}{3} + (5 + E[X]) \cdot \frac{1}{3}$$

(All doors have equal probability)

So we can find  $E[X] = 10$ .

### 3. Hiring problem

#### 3.1

There is a trade off because if we choose k as bigger value we probably will see the best person in this range (1..k). But we cannot hire this (best) person. However if we choose small k we will find the best score of these k person then we start to evaluate the others. But it does not mean that smaller k values have larger chance to find the best person. Because according to the algorithm we should select the first person that has higher score. It does not mean we find the best one. For instance if the selected person index is j ( $k < j < n$ ), candidates which have indexes between j and n could have higher score.

Selection of the suitable k is an important issue. It also depends on input.

#### 3.2

In below equation i is assigned to k+1 instead of 1. Because we do not select any candidate in first phase which covers indexes from 1 to k. So the probability of them are zero.

$$\Pr\{S\} = \sum_{i=k+1}^n \Pr\{S_i\}$$

If it is possible to select k as (index\_of\_best\_candidate)-1, according to the algorithm in the second phase we should select the first higher score which is the best person in this case.

#### 3.3

$S_i = B_i \cap O_i$  means that in second phase (after k candidate is rejected) we should not select any person until we encounter with the best person which has the index of i. So

$$\Pr\{S_i\} = \Pr\{B_i\} \cap \Pr\{O_i\} = \Pr\{B_i\} \cdot \Pr\{O_i\}$$

#### 3.4

$$\Pr\{S\} = \frac{k}{n} \sum_{i=k+1}^n \frac{1}{i}$$

We can rewrite the equation as

$$= \frac{k}{n} \left( \sum_{i=1}^{n-1} \frac{1}{i} - \sum_{i=1}^k \frac{1}{i} \right) \quad (\text{Eq. a})$$

If we use the lemma which is given for harmonic series such as

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{n} = \sum_{k=1}^n \frac{1}{k}$$

Lemma:  $\int_1^{k+1} \frac{1}{x} dx = \ln(k+1) < \sum_{n=1}^k \frac{1}{n}$  Then, Eq. a becomes

$$= \frac{k}{n} (\ln(n) - \ln(k))$$

In order to maximize the bound we should differentiate this equation and make equal to zero

$$= \frac{d}{dk} \left( \frac{k}{n} (\ln(n) - \ln(k)) \right) = 0$$

$$= \frac{1}{n} (\ln(n) - \ln(k) - 1) = 0$$

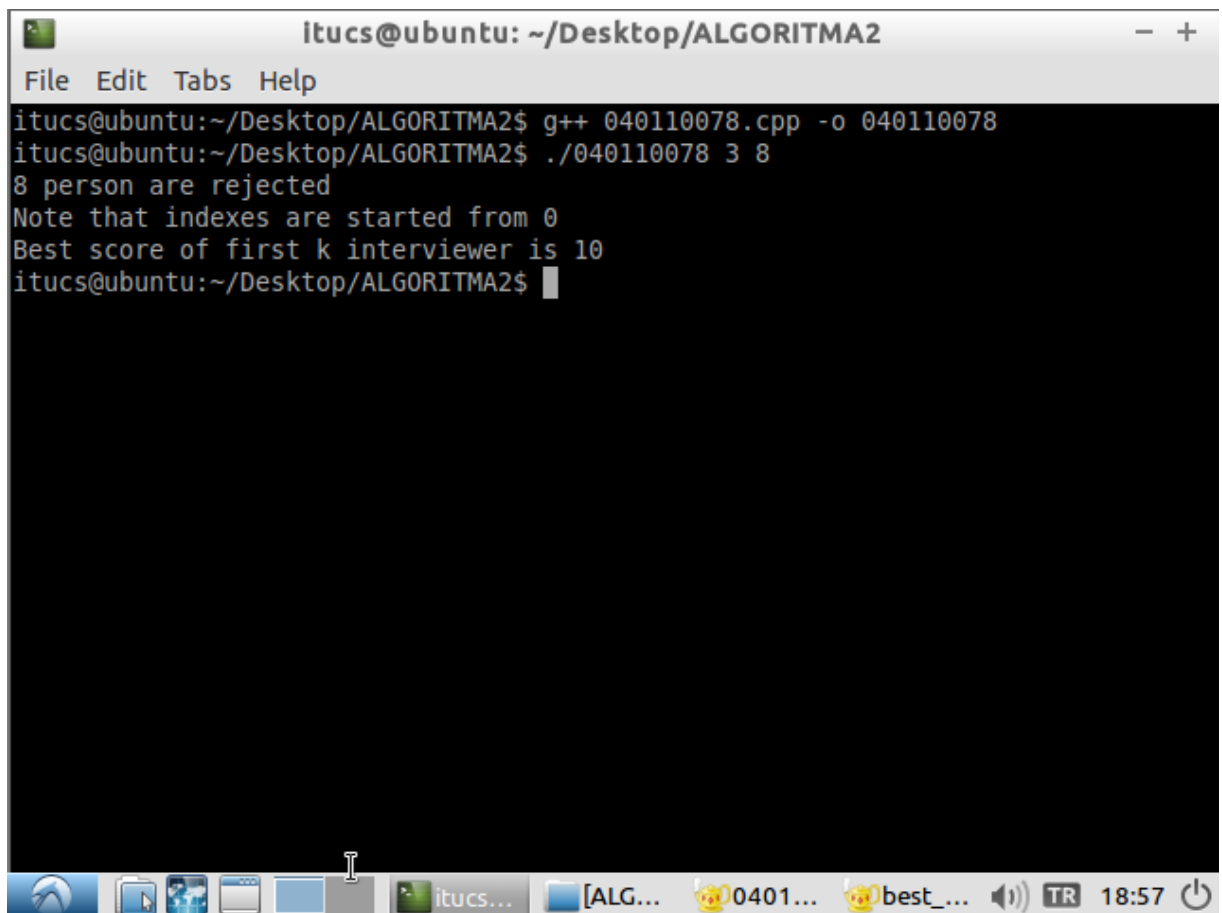
$$= (\ln(n) - \ln(k) - 1) = 0$$

$$= \ln(k) = \ln(n) - 1$$

$$= k = \frac{e}{n}$$

## Part B. Implementation and Report

### 1. Implementation



```
itucs@ubuntu: ~/Desktop/ALGORITMA2
File Edit Tabs Help
itucs@ubuntu:~/Desktop/ALGORITMA2$ g++ 040110078.cpp -o 040110078
itucs@ubuntu:~/Desktop/ALGORITMA2$ ./040110078 3 8
8 person are rejected
Note that indexes are started from 0
Best score of first k interviewer is 10
itucs@ubuntu:~/Desktop/ALGORITMA2$
```

Screenshot for the algorithm is given above. The code is compiled and executed on Ubuntu 14.04 via VM VirtualBox.

## 2. Report of Implementation

According to the algorithm first applicant is interviewed then the highest score of them is founded and assigned to the 'bestofk' variable. In the second part the person who has better score than bestofk is hired.

**Note that:** In below tables the best applicant does not mean it is hired. May be the best applicant exist in first k applicant in terms of his score. But we reject them. If there is no one that his score is better than bestofk we should accept the last applicant (last element of the array).

### For set A

k	2	N/e	8
The best applicant index	3	7	9
Applicant score	9	10	5
Running time	0.000048	0.000000	0.000059

### For set B

k	2	N/e	8
The best applicant index	9	9	9
Applicant score	6	6	6
Running time	0.000065	0.000000	0.000000

**For set C**

<b>k</b>	<b>2</b>	<b>N/e</b>	<b>8</b>
<b>The best applicant index</b>	<b>5</b>	<b>5</b>	<b>9</b>
<b>Applicant score</b>	<b>10</b>	<b>10</b>	<b>8</b>
<b>Running time</b>	<b>0.000088</b>	<b>0.000000</b>	<b>0.000062</b>