كليـــة .....

كونترول الفرقة .....

العـــام الجامعـــى 2019 / 2020 – دور مايـــو

الغلاف الخارجى للبحث

أولاً: البيانات الخاصة بالطالب			
الفرقة الدراسية	الثانية	التخصص	عام
اسم القسم	عام		
اسم المقرر	خوارزميات		
استاذ المقرر	د.مروة عبد الفتاح		

ثانياً: البيانات الخاصة بالبحث			
عنوان البحث	Task-7 : Phone Numbers		
طبيعة المشاركة	بحث فردى نعم		بحث جماعى
ارسال البحث	بواسطة البريد الالكتروني		
اسماء الطلاب المشاركين فى البحث (يكتب الاسم رباعيا)	م	الاسم رباعى	رقم الجلوس	الرقم القومى
	1	سيد معتز محمد سيد	2408	29903140103453
	2			
	3			
	4			
	5			
تاريخ الإرسال	1 / 6 / 2020			

ثالثاً: البيانات الخاصة بالكونترول			
النتيجة	ناجح		راسب
أعضاء لجنة تقييم البحث		الاسماء	التوقيع
	1		
	2		
	3		

فى حالة عدم قبول البحث يرجى ذكر الأسباب	- ..... - ..... - ..... - .....

1. Problem Statement

a.

You're the manager for the phone company on board a new space station, and you're trying to assign numbers to various services. To increase dialing speed, you want a call to be placed as soon as a valid number is dialed. For example, if 911 is the number for emergency services, then entering those three digits starts the call immediately, so you can't assign the number 9113 to some other service, since anyone trying to dial it would end up calling 911. There are n phone services. You have statistics for each service from other space stations; you know that the ith service is used on average fi times per year. Given f1…..fn, you want to assign phone numbers to the services (using only the digits 0–9) so that services which are used more often will have shorter numbers. In particular, you want to minimize the average number of digits dialed (with the average taken over all calls to a phone service)

b.

I need to generate non-repeating numbers for different services to increase dialing speed so that at the same time I need to tradeoff between the number of digits of each number and the frequency of the service that means the number of times that people use this service, the more frequency increase, the more number of digit of service number decrease, so I used a built-in function rand() to generate random numbers with the simple constraint that numbers store in an array of integer but before store it, I should check if this number stored before or not to make sure that there is no repeating numbers then I used insertion sort algorithm to sort Frequency from bigger to smaller and one time more services number from smaller to bigger, finally. now I can assign services numbers to its services

c.

One of big Problems that the services which are used most often will have shorter phone numbers

2. Design: Pseudocode

```
struct services contain
        Char ServicesName[20]
        int ServicesFreq
        int ServicesNumber
End
Algorithm sort_services
Input:
ServicesName[20]
ServicesFreq
Output:
Algorithm GenerateRandomNumbers
Begin
Read <- NumOfServices
New services s
Loop I from 0 to NumOfServices-1
Write -> service I name:
Read <- ServicesName
Write -> service i freq:
Read <- ServicesFreq
End loop
Algorithm Sort_services ( struct s)
     Loop I = 0 to n-1
              If (A[i] < A [i+1])
                 A[i] ← A [i+1]
     End loop
Algorithm GenerateRandomNumbers ( struct s)
        Loop I = 0 to n-1
              If (i < 20)
                   A[i] ← rand()%100+1
              Else if ( i < 50)
                   A[i] ← rand()%1000+1
              Else if ( i < 50)
                   A[i] ← rand()%10000+1
              Else
                   A[i] ← rand()%100000+1
        End loop
Algorithm AccessGeneratedRandomNumbers( struct s)
        Loop I = 0 to n-1
              If (A[i] < A [i+1])
                 A[i] ← A [i+1]
        End loop
      Write → name of services
      Write → frequency of services
      Write → number of services
End
```

3. Analysis and Compute the time complexity

I've 6 functions at my solutions

1. NumberOfServices

Basic Operation:

Read ← number of services

Time Complexity:

$$C(n)^1 = 1$$

2. ReadData

Basic Operation:

Read ← name of services

Read ← frequency of services

Time Complexity:

$$C(n)^1 = \sum_{i=0}^{n-1}(\mathrm{n} - 1 - 0 + 1) = n$$

3. sortServices

Basic Operation:

```
If (A[i] < A [i+1])
    A[i] ← A [i+1]
```

Time Complexity:

$$C(n)^1 = \sum_{i=0}^{n-1}(\mathrm{n} - 1 - 0 + 1) \sum_{j=0}^{n-1}(\mathrm{n} - 1 - 0 + 1)$$

$$= \sum_{i=0}^{n-1}(\mathrm{n} - 1 - 0 + 1)\ (\mathrm{n})$$

$$= (n - 1 - 0 + 1\ )\ (\mathrm{n})$$

$$= (n\ )(\mathrm{n})$$

$$= n^2$$

4. GenerateRandomNumbers

Basic Operation:

```
If (i < 20)
    A[i] ← rand()%100+1
Else if ( i < 50)
    A[i] ← rand()%1000+1
Else if ( i < 50)
    A[i] ← rand()%10000+1
Else
    A[i] ← rand()%100000+1
```

Time Complexity:

$$C(n)^1 = \sum_{i=0}^{n-1}(\mathrm{n} - 1 - 0 + 1) = n$$

5. AccessGeneratedRandomNumbers

Basic Operation:

If (A[i] < A [i+1])
A[i] ← A [i+1]

Time Complexity:

$$C(n)^1 = \sum_{i=0}^{n-1}(\mathrm{n} - 1 - 0 + 1) \sum_{j=0}^{n-1}(\mathrm{n} - 1 - 0 + 1)$$

$$= \sum_{i=0}^{n-1}(\mathrm{n} - 1 - 0 + 1)\ (\mathrm{n})$$

$$= (n - 1 - 0 + 1\ )\ (\mathrm{n})$$

$$= (n\ )(\mathrm{n})$$

$$= n^2$$

6. PrintServices

Basic Operation:

Write → name of services
Write → frequency of services
Write → number of services

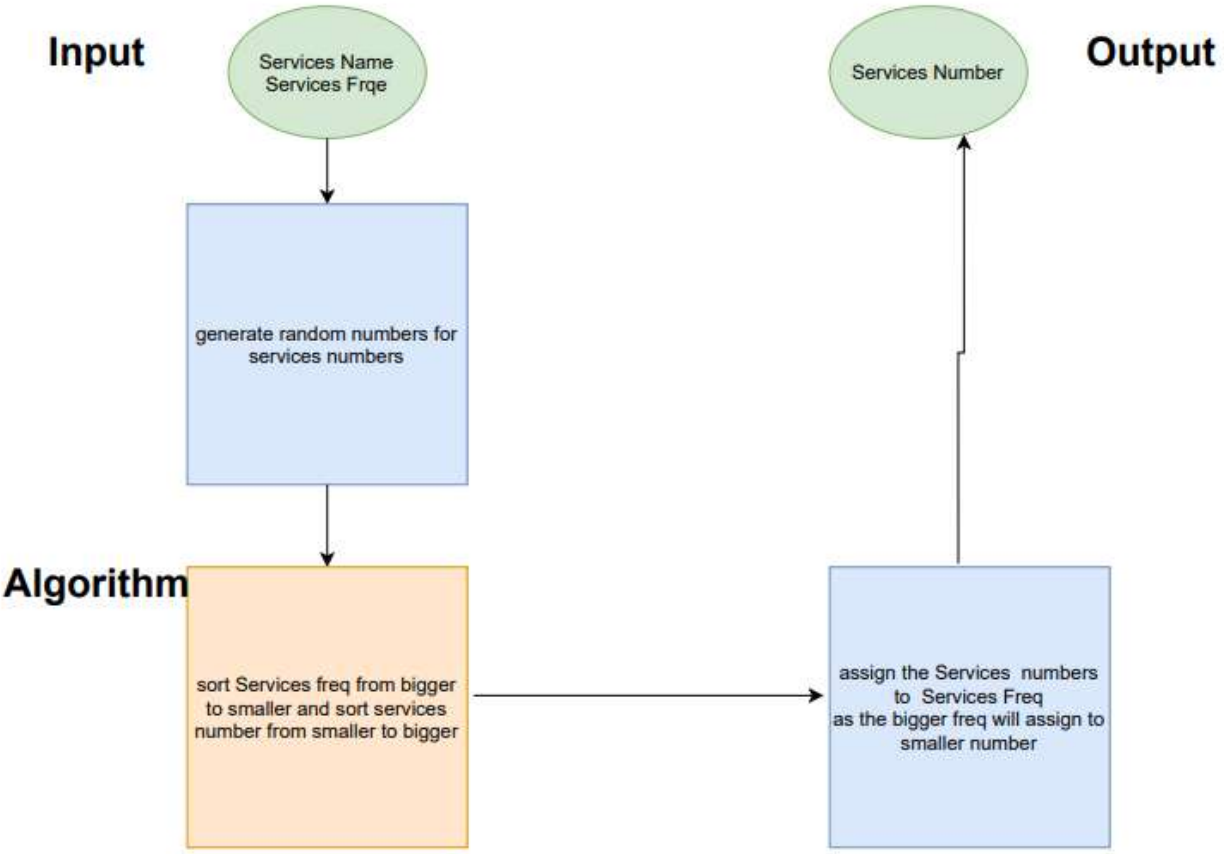
Time Complexity:

$$C(n)^1 = \sum_{i=0}^{n-1}(\mathrm{n} - 1 - 0 + 1) = n$$

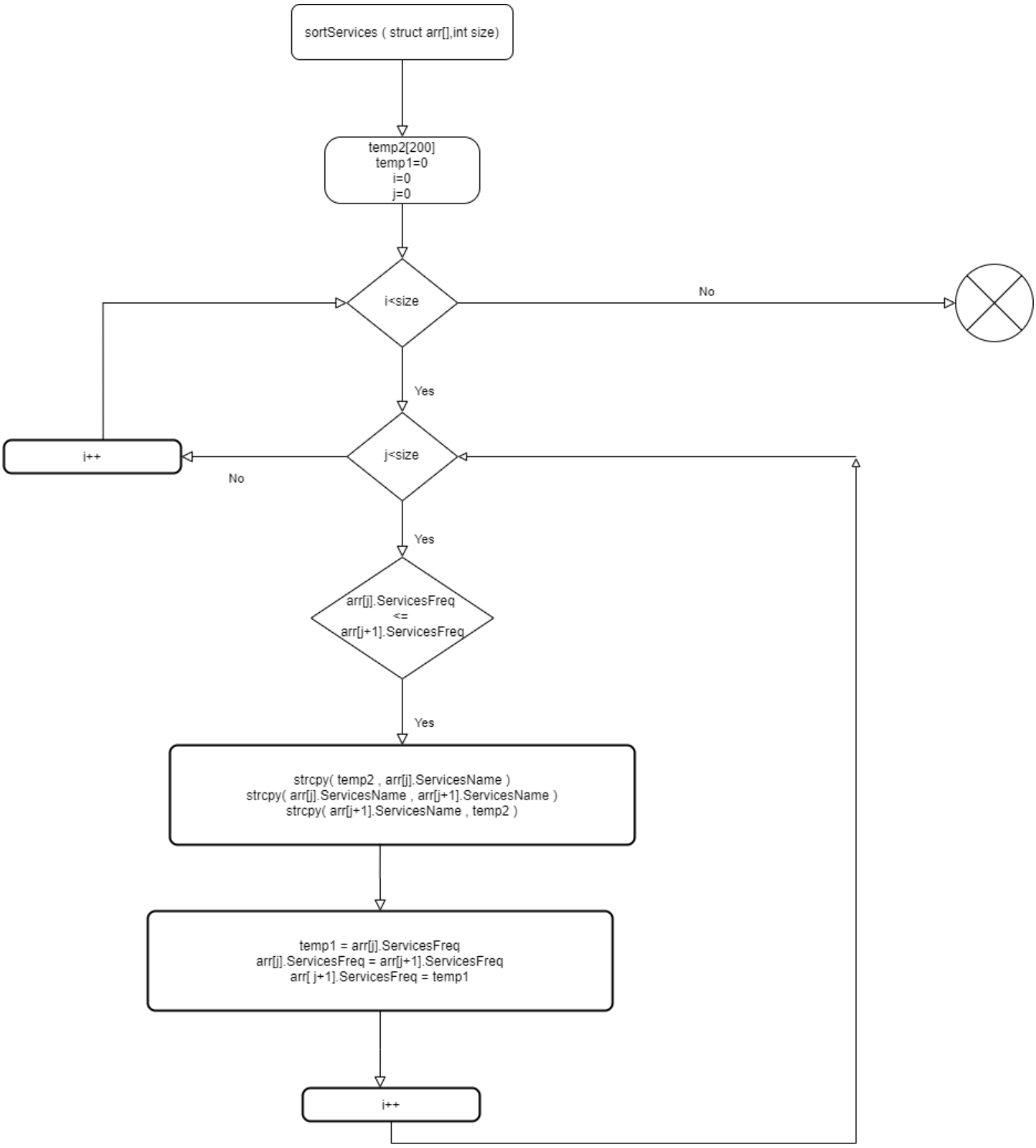
Based on the above, the time complexity for the complex function is n ^ 2, so the time complexity for the full program is n ^ 2

4. System Implementation and results

a. Block diagram



Flowchart for main Algorithm


sortServices (struct arr[],int size)
temp2[200]
temp1=0
i=0
j=0
i<size
No
Yes
j<size
No
i++
Yes
arr[j].ServicesFreq
<=
arr[j+1].ServicesFreq
Yes
strcpy(temp2 , arr[j].ServicesName)
strcpy(arr[j].ServicesName , arr[j+1].ServicesName)
strcpy(arr[j+1].ServicesName , temp2)
temp1 = arr[j].ServicesFreq
arr[j].ServicesFreq = arr[j+1].ServicesFreq
arr[j+1].ServicesFreq = temp1
i++

b. System results are sorted random numbers from smallest to bigger assign to sort input frequency from bigger to smaller

Ex.

Frequency	Number
485611	8
31284	25
2498	68
248	99
67	160
0	888

And its performance we can act it by time complexity Cn (N^2)

c.

C:\Users\Moataaz\Desktop\algorithms\fifthtry\bin\Debug\fifthtry.exe

Services number	Services name	Services freq	New Services Number
1	¢B@	2003291328	1
2	erh	1857421840	2
3	regf	1816255659	5
4	dfhb	1353464839	6
5	rf	1208035980	8
6	eh	628776235	10
7	sfg	16894516	25
8	erh	16777472	28
9	trh	12987608	28
10	db	12976320	36
11	d	6835808	37
12	erh	6541312	43
13	erfb	5656418	46
14	rgfv	4651230	59
15	dsv	2389456	62
16	erv	984651	63
17	sgv	984651	65
18	5tef	651748	79
19	rth	645123	82
20	gf	485623	92
21	gsd	189489	96
22	sdv	154894	142
23	reg	98456	154
24	f	96512	254
25	6hyt	94865	293
26	f	89465	300
27	regf	86523	323
28	d	65123	334
29	fc	65123	383
30	fc	65123	392
31	sv	65123	422
32	sfv	65123	448
33	ed	32148	539
34	efd	23554	605
35	eth	9465	665
36	fb	6123	668
37	fv	6123	674
38	fc	5163	704
39	gsf	4984	712
40	fb	4856	717
41	sg	2841	719
42	sg	1982	727

```
C:\Users\[ ConceptDraw DIAGRAM - [Concept1 - Page1] \fifthtry.exe
40              fb              4856            717
41              sg              2841            719
42              sg              1982            727
43              rg              612             772
44              sdg             489             812
45              ac              321             869
46              as              321             870
47              /79             4               895
48              tbgd            2               896
49              ergf            0               903
50              erh             0               913
51              ger             0               5548

Process returned 0 (0x0)   execution time : 110.062 s
Press any key to continue.
```

5. Appendix

a. Source Code

b. PowerPoint++record

https://drive.google.com/drive/folders/1D7vMNd9pSeB_WPZOXnsgKYxUv4Njeh1H?usp=sharing