KLE Technological University
Creating Value
Leveraging Knowledge

Earlier known as
B. V. B. College of Engineering & Technology

School of Computer Science and Engineering

MINOR EXAM II - SOLUTIONS

Course : Data Structures and Algorithms	SRN	0	1	F	E			B	C	S			

Course Code: 19ECSC201
Semester: III
Date of Exam: 12 Nov 2019
Duration: 1 hour 15 minutes

Men are from Mars, Women are from Venus, Algorithms are from Hell. Probably.

Those mistakes you were said not to do, now is the time!

QUESTION 01

1a You could be Quick. But are you with Quick? **04 marks**

Apply Quick Sort on the following given numbers: (the subscripts to the numbers indicate the occurrence order and all the given 4's have same value.) **L3** **CLO3** **06 mins**

4_a 2 4_b 3 4_c

Quick Sort Tracing:

4_a 2 4_b 3 4_c

P i j

Swap i and j

4_a 2 4_c 3 4_b

P j i

Swap P and j

3 2 4_c 4_a 4_b

4_a has found its home

3 2 4_c

P j i

Swap P and j

2 3 4_c

3 has found its home

Remaining left are all single keys. The sorted order is 2 3 4_c 4_a 4_b

Quick sort is NOT stable.

1b **881 is Prime. So is 1009. And 1013.**

Use Rabin-Karp algorithm to find the first occurence of the pattern in the given text:

TEXT: BAZZINGA

PATTERN: ZINGA

06 marks
(04 + 02)
L3
CLO3
08 mins

Let us consider prime number as 11.

Hash(ZINGA) $= 26 * 11^0 + 9 * 11^1 + 14 * 11^2 + 7 * 11^3 + 1 * 11^4$
$= 26 + 99 + 1694 + 9317 + 14641$
$= 25777$

Iteration 01:

BAZZINGA
ZINGA

Hash(BAZZI) $= 2 * 11^0 + 1 * 11^1 + 26 * 11^2 + 26 * 11^3 + 9 * 11^4$
$= 2 + 11 + 3146 + 34606 + 131769$
$= 169534$

Hash values do not match

Iteration 02:

BAZZINGA
 ZINGA

RollingHash(AZZIN) $= (169534 – 2) / 11 + 14 * 11^4$
$= 15412 + 204974$
$= 220386$

Hash values do not match

Iteration 03:

BAZZINGA
  ZINGA

RollingHash (ZZING) $= (220386 – 1) / 11 + 7 * 11^4$
$= 20035 + 102487$
$= 122522$

Hash values do not match

Iteration 04:

BAZZINGA
   ZINGA

RollingHash(ZINGA) = (122522 – 26) / 11 + 1 * 11^4
= 11136 + 14641
= 25777

Hash match.
We now compare the characters one by one. Match found.

Explain how Rabin-Karp algorithm builds and improves over the Brute Force String Search.
It's basically a Brute force string match. We compare hashes rather than individual characters in the string and compare individual only when there is a hash match.

1c **Universe is a Graph and Graph has Stories.**
Answer the following plots:
i. Explain the role of min-heap data structure in Dijkstra's algorithm.
Dijkstra's process needs a priority queue implementation to select a minimum weighted edge during edge relaxation process. The process is efficient if priority queue is implemented as min-heap

10 marks
(05 * 02)
L3
CLO4
15 mins

ii. Implement the function for the given API from union-find:
void union (int a[ ], int i, int j)
Assume size of array 'n' is global.

```
void union (int a[ ], int i, int j)
{
   int temp = a[i];
   int x;
   for( x  0; x < n;  x++) {
     if(a[x] == temp)
         a[x] = a[j];
   }
}
```

iii. Six Degrees of Separation - Everyone and everything is six or fewer steps away. A chain of "a friend of a friend" statements can be made to connect any two people in world in a maximum of six steps. Comment on the role of shortest path and spanning tree algorithms in achieving the same. (yes, you could be related to Rajkumar Bala Dev Singh in 6 or fewer steps)
The technique is about establishing minimum hops to reach from a source to

destination. Basically a variant of spanning tree but more of traversal than shortest path or spanning tree.

iv. The Sorting Hat from the movie series Harry Potter is a hat that magically determines which of the four houses (Gryffindor, Hufflepuff, Ravenclaw, Slytherin) does every student belong to. Why is the hat called 'Sorting Hat' when all it does is classification? Explain your understanding.
Sorting has numerous applications. Though the Sorting Hat does classification, it is sorting students into four different houses.

v. Explain the meme. What is the nerd hinting at?

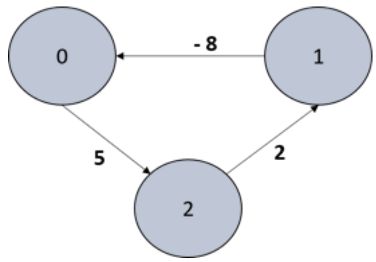
The question asked is efficiency analysis of Kurukals algorithm.

QUESTION 02

2a **Negative? Bellman-Ford Saves the Day**

Apply Bellman-Ford algorithm on given graph with source vertex as 0.

04 marks
L2
CLO3
04 mins



The graph has a negative weight cycle (5 + 2 - 8). Bellaman-Ford returns that graph has negative weight cycle and not the shortest path.

2b **How Sorted Are You?**

i) Which sorting algorithm of that you have studied, will take least time when all elements of input array are identical?

Insertion Sort - Ω (n)

ii) You are given with two sorted lists of size m and n respectively. What is the number of comparisons needed in the worst case by the merge phase of the merge sort algorithm?

To merge two lists of size m and n, we need to do m+n-1 comparisons in worst case. Since we need to merge 2 at a time, the optimal strategy would be to take smallest size lists first. The reason for picking smallest two items is to carry minimum items for repetition in merging.

06 marks
(03 * 02)
L3
CLO4
10 mins

iii) Complete the function described below:

```
Function:    returnParent
Description: for the supplied i, return its parent from
             array H, where H is a max heap
Input:       max-heap H and i
Output:      index of i's parent, -1 otherwise
int returnParent(H[1 . . .n], i)
{
   if(i == 1)
      return -1;
   else
      return i/2;
}
```

2c **A Mess, A Mesh, A Mix.**

i. The average case recurrence relation for quick sort is given by:

$$T(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ 1/n * \sum_{s=0}^{n-1} [T(s) + T(n-1-s) + (n+1)] & \text{otherwise} \end{cases}$$

Explain the relation, where s is the split position and n is input size.

The partition can happen in each position s with same probability 1/n. The n+1 indicates the number of comparisons and other terms represent the split and remaining items.

Note: errata: when n = 1, T(n) = 0.

10 marks
(05 * 02)
L3
CLO3
16 mins

ii. Construct the prefix table (π-table) for the pattern

POP

Substring	Proper Prefixes	Proper Suffixes	Π-value
P	NIL	NIL	0
PO	P	O	0
POP	P, PO	P, OP	1

Prefix Table:

char	P	O	P
index	0	1	2
value	0	0	1

iii. Given below is a matrix which cumulates assortment of knowledge into a 4x4 grid. Convert this into a graph whose traversal/application can bring out the cumulated knowledge.

S	T	E	P
T	I	M	E
E	M	I	T
P	E	T	S

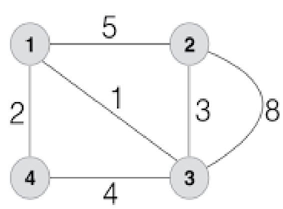
This would be an interesting graph (transition diagram) with several variants.

Basically, the intuition is to have 4 nodes, connecting with several paths denoted by alphabtes as seen in grid above. Draw a graph such that any traversal leads to above given words: STEP, TIME, EMIT and PETS.

iv. State two differences between Prim's and Kruskals Algorithms.

Kruskal's	**Prim's**
Grows with minimum cost edge	Grows with minimum cost vertes
If we stop the algorithm in the middle, we can get a disconnected tree or forest	If we stop algorithm in the middle, prims always generates a connected tree
Need to give attention to cycle check	Need not give attention to cycle check
Edge selection is not based on pevious step	Spans from one vertex to another

v. The graph given below needs to be supplied to Warshall's algorithm.Write the input matrix for the given weighted graph.



	1	**2**	**3**	**4**
1	0	1	1	1
2	1	0	1	0
3	1	1	0	1
4	1	0	1	0

QUESTION 03

3a **Bad Table isn't that Bad and Good Table isn't that Good!** **04 marks**

How many character comparisons will the Boyer-Moore algorithm make in searching for the pattern: 10000 in the binary text of one thousand zeros? **L3** **CLO4** **06 mins**

Bad-Symbol Table:

c	0	1
T(c)	1	4

Good Suffix Table

k	pattern	d2
1	10000	3
2	10000	2
3	10000	1
4	10000	5

On each of its trials, the algorithm will make four successful and one unsuccessful comparison and then shift the pattern by the maximum of d1 = max{t1(0) − 4, 1} = 1 and d2 = 5, i.e., by 5 characters to the right. The total number of character comparisons will be = 5 * 200 = 1000.

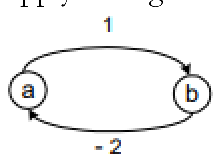
3b **You Cannot Avoid the Floyd** **06 marks**

Write Floyds Algorithm to compute all pair shortest path problem. **(03 + 03)**

Apply the algorithm on the given graph below: **L2**

CLO3

08 mins



ALGORITHM Floyd (W[1..n,1..n])
// Implements Floyd's algorithm for all pair shortest path problem
// Input: The weight matrix W of the graph with no negative length cycle
// Output: The distance matrix of the shortest path's lengths
D ← W
for k ← 1 to n do
    for i ← 1 to n do
        for j ← 1 to n do
            D [i, j] ← min {D[i, j], D[i, k] +D[k, j]}
return D

$$D^{(0)} = \begin{bmatrix} 0 & 1 \\ -2 & 0 \end{bmatrix} \qquad D^{(1)} = \begin{bmatrix} 0 & 1 \\ -2 & -1 \end{bmatrix} \qquad D^{(2)} = \begin{bmatrix} -1 & 0 \\ -3 & -2 \end{bmatrix}$$

None of the four elements of the last matrix gives the correct value of the shortest path, which is, in fact, $-\infty$ because repeating the cycle enough times makes the length of a path arbitrarily small. Floyd's algorithm can be used for detecting negative-length cycles, but the algorithm should be stopped as soon as it generates a matrix with a negative element on its main diagonal.

3c Sorting Towards Searching

In insertion sort, we first search for the place of insertion using linear search and then insert at found position. Write an algorithm for binary-insertion sort which uses binary search to search for the location to perform the insertion.

Write your efficiency analysis of binary-insertion sort. How does it perform over the traditional Insertion sort?

10 marks
(07 + 03)
L3
CLO4
16 mins

```
ALGORITHM binarySearch(int a[], int item, int low, int high)
    if (high <= low)
        return (item > a[low])?  (low + 1): low
    mid ← (low + high)/2
    if item = a[mid]
        return mid+1
    if item > a[mid]
        return binarySearch(a, item, mid+1, high)
    return binarySearch(a, item, low, mid-1)

ALGORITHM insertionSort(int a[], int n)
    for i from 1 to n
        j ← i - 1
        selected ← a[i]
        loc ← binarySearch(a, selected, 0, j)

        while j >= loc
            a[j+1] ← a[j]
            j ← j - 1
        a[j+1] ← selected
```

The algorithm as a whole still has a running worst case running time of $O(n^2)$ because of the series of swaps required for each insertion.