COSC 2341 Data Structures and Algorithms Exam 2

What is allowed to use during the exam:

- Textbook and notes (your own and/or the instructor's)
- Your own memory sheet(s)
- Your own code from the labs
- Windows calculator or pocket calculator

What is not allowed to use:

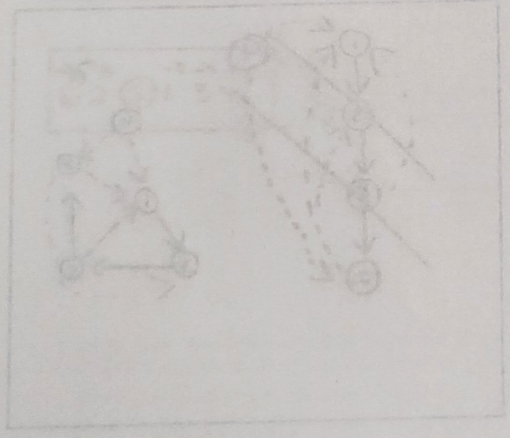
- Any web browser (except for using a web-based IDE for programming)
- Any input from other people

Please show all the steps in your solution. Credit is awarded for complete solutions, not just for the final answers. The final answers by themselves (especially when wrong!) will bring little credit.

For the coding problem(s) include in your report screenshots of both code and output.

I, the undersigned, understand the instructions above, and I will abide by them. I will not try to seek, use or provide any dishonest help.

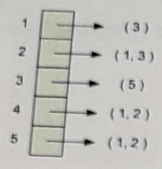
Name (Print!): Edwin (Sonny) Sparks Signature: Edwin (Sonny) Sparks

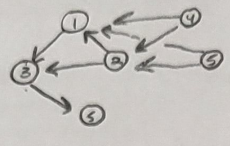


COSC 2341 Data Structures and Algorithms Exam 2

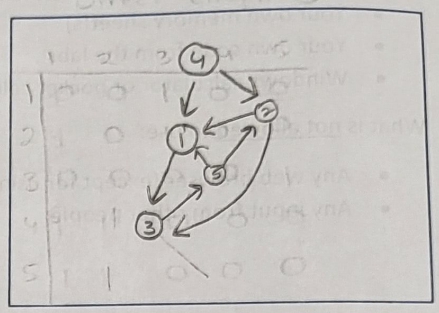
Pencil and paper problems

1] (20 pts.) A directed graph is represented by this adjacency list:



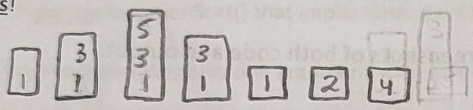


(5 pts.) Draw the graph here:



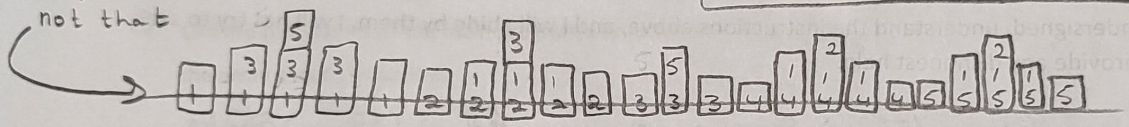
Why is nothing pointing to 4?

(10 pts.) Apply the DFS traversal starting at node 1, showing the stack of nodes at each step. Keep track of both indices!



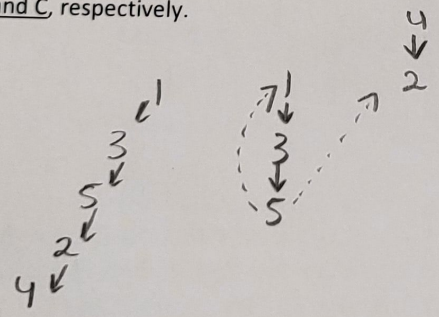
1 3 5 2 4 DFS Traversal

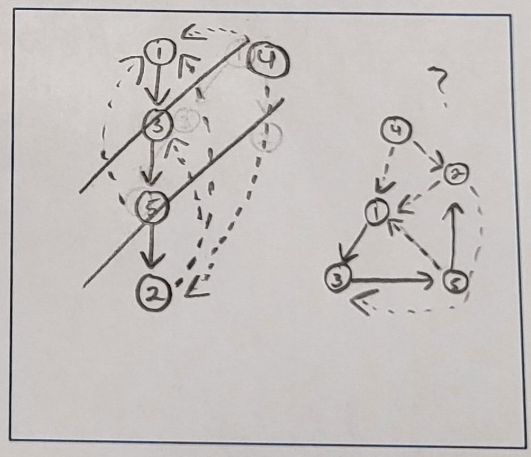
LoL, Def. not that



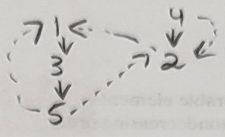
(3 pts.) Draw the traversal tree inside the rectangle on the right:

(2 pts.) On the same drawing show the forward, back and cross edges with dashed lines. Mark them clearly with the letters F, B and C, respectively.



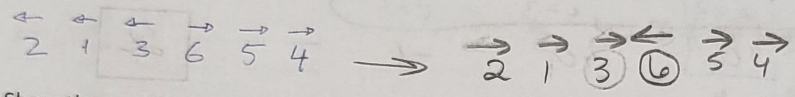


Extra-credit (5 pts.) Using the indices found in the previous problem, write/draw the topological sorted order for the graph:



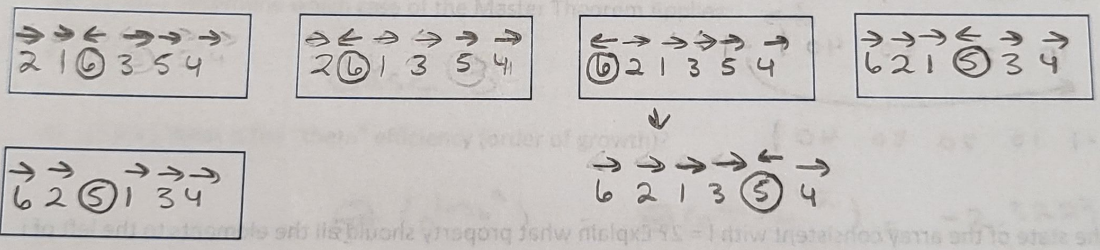
What do you conclude? A: There is no route to 4 ?

2] (20 pts.) When applying the Johnson-Trotter algorithm for permutations of size n = 6, the following permutation was reached:



Show the next five permutations.

- Clearly mark (circle, underline, bold, highlight, etc.) the LARGEST MOBILE ELEMENT in each permutation, starting with the one given above! 6
- Do not forget the arrows!



3] (20 pts.) The array shown below arises during the application of the **Insertion Sort** algorithm, at the point when **i** is given the value 2:

```
ALGORITHM InsertionSort(A[0..n − 1])
    //Sorts a given array by insertion sort
    //Input: An array A[0..n − 1] of n orderable elements
    //Output: Array A[0..n − 1] sorted in nondecreasing order
    for i ← 1 to n − 1 do
        v ← A[i]
        j ← i − 1
        while j ≥ 0 and A[j] > v do
            A[j + 1] ← A[j]
            j ← j − 1
        A[j + 1] ← v
```

No idea what this is

j + 1 = 3

i = 0	i = 1	i = 2	i = 3	i = 4
20	30	10	40	-1

v: 2

0 1 2 3 4

i = 1: 20 | 30 10 40 -1

i = 2: 20 30 | 10 40 -1

i = 3: 10 20 30 | 40 -1

i = 4: 10 20 30 40 | -1

i = 5: -1 10 20 30 40 |

(a) Is the state of the array consistent with i = 2? Explain what property should all the elements to the left of i have in the Insertion Sort algorithm: They Have been Seen / visited

(b) What values are now given to v = 30 and j = 40?

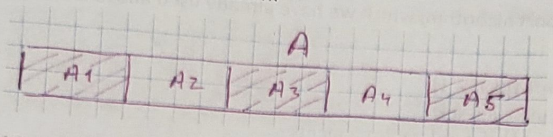
Draw above the array, the variable v, and the indices i and j after each iteration of the *while* loop. Stop when i is given the next value, 3 – do not finish the algorithm!!

- Clearly show each swap with arrows!

~~(c) Is the state of the array consistent with i = 2? Explain what property should all the elements to the left of i have in the Insertion Sort algorithm: _____~~

~~_____~~

4] (15 pts.) A certain divide-and-conquer algorithm works by dividing the original array A into five sub-arrays A1, A2, A3, A4, A5 as shown:



A1 ... A5 all have a size that is exactly one fifth that of the original array A. Assume that the number of elements n is always a multiple of 5.

The algorithm is then called recursively to solve only the three sub-arrays A1, A3 and A5 (shaded above).

Putting together the solutions to the sub-arrays requires **42·sqrt(n)** basic operations. $n^{1/2}$

a) (5 pts.) Identify the three parameters needed in the Master Theorem:

$T(n) = 42T(\frac{n}{5}) + f(n)^{1/2}$

$42\,n^{1/2}$

a = 42

b = 1/5

d = 1/2

$42 > 0.447214$

b) (5 pts.) Determine which case of the Master Theorem Applies:

Case 3

c) (5 pts.) What is the "theta" efficiency (order of growth)?

$\Theta(n^{\log_b a}) = \Theta(n^{\log_{1/5} 42}) = -2.32234$

$\Theta\ 1/n^{2.322}$

5×10^{-9} Sec

⌂ **Extra-credit** (5 pts.) If in the problem above each basic operation takes 5 ns (nanosec.), what is the estimated running time of the algorithm for and array with n = 1 billion (10^9) elements? Use only the theta found above. Can the solution be found practically in real-life with this algorithm?

Yes, its practical as n get big run time lowers

$1/1000000000^{2.322} \cdot 5 = 5.847 \times 10^{-14}$ Sec

.0000000...

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Midterm_2 - main.c

Midterm_2

Midterm_2 | Debug

Project

Midterm_2 E:\Tarleton Fall 21'\D
External Libraries
Scratches and Consoles

main.c

Indexing...

```
// Sonny Sparks
// Data Structures
// Midterm 2
// October 27, 2021

#include <stdio.h>
#define N 10
int A[N]={10,20,-10,-20,1,2,3,4,100,-100};
//////////////////////////////////////////////////      //Function Start
void InsertionSort(int n) {
    for (int i = 1; i <= n-1; i++) {
        int v = A[i];                                   //Master loop (i)
        int j = i - 1;
        ///////////////
            for(int z=0;z<=n-1;z++)
                printf( _Format: "\t%d", A[z]);          //Iteration Print
            printf( _Format: "\n");
        /////////////
        while (j >= 0 && A[j] > v){
            A[j + 1] = A[j];                            //Insertion loop (j)
            j = j - 1;
        }
        A[j + 1] = v;
    }
    printf( _Format: "The Sorted array is : \n");       //Final Iteration Print
    for(int h=0;h<=n-1;h++)
        printf( _Format: "\t%d", A[h]);
}
////////////////////////////////////////////////
int main() {
    InsertionSort(N);                                   //Call to Function
}
```

Run: Midterm_2

```
"E:\Tarleton Fall 21'\Data Structures Lab\Midterm_2\cmake-build-debug\Midterm_2.exe"
        10      20      -10     -20     1       2       3       4       100     -100
        10      20      -10     -20     1       2       3       4       100     -100
        -10     10      20      -20     1       2       3       4       100     -100
        -20     -10     10      20      1       2       3       4       100     -100
        -20     -10     1       10      20      2       3       4       100     -100
        -20     -10     1       2       10      20      3       4       100     -100
        -20     -10     1       2       3       10      20      4       100     -100
        -20     -10     1       2       3       4       10      20      100     -100
        -20     -10     1       2       3       4       10      20      100     -100
The Sorted array is :
        -100    -20     -10     1       2       3       4       10      20      100
Process finished with exit code 0
```

Database

Run