Algorithms and Data Structures

24.02.2023

Exercise 1
Given the following array of integer values, perform the first step of quicksort to sort the array in ascending order; then, from the initial array, generate the right and the left partitions.

```
10  8  9  7  2  6  4  3  1  4
```

Report 3 integer values: The pivot selected on the original array, the pivot you would select on the left partition generated from the original array, and the pivot you would select on the right partition generated (again) from the original array. No other symbols must be included in the response. This is an example of the response format: 13 1 10

```
### Quick sort
Solution format 1:
---------------------------------------------
Initial array: 10  8  9  7  2  6  4  3  1  4
---------------------------------------------
Re l= 0;r= 2:   1  3  2
Re l= 0;r= 0:   1
Re l= 0;r= 0:         3
Re l= 0;r= 2:               9  6  4  8 10  7
Re l= 4;r= 5:               4  6
Re l= 4;r= 4:               4
Re l= 4;r= 4:
Re l= 4;r= 5:                        8 10  9
Re l= 7;r= 7:                        8
Re l= 7;r= 7:                             10
---------------------------------------------
Solution format 2:
---------------------------------------------
Input array:   [0]10 [1]8 [2]9 [3]7 [4]2 [5]6 [6]4 [7]3 [8]1 [9]4
Pivot:         [9]4
Swaps:         [0<->8]10<->1  - [1<->7]8<->3  - [2<->4]9<->2  - [3<->9]7<->4
Output array:  [0]1 [1]3 [2]2  - [3]4  - [4]9 [5]6 [6]4 [7]8 [8]10 [9]7
---------------------------------------------
Input array:   [0]1 [1]3 [2]2  - [4]9 [5]6 [6]4 [7]8 [8]10 [9]7
Pivot:         [2]2  - [9]7
Swaps:         [1<->2]3<->2  - [4<->6]9<->4  - [6<->9]9<->7
Output array:  [0]1  - [1]2  - [2]3  - [4]4 [5]6  - [6]7  - [7]8 [8]10 [9]9
---------------------------------------------
Input array:   [4]4 [5]6  - [7]8 [8]10 [9]9
Pivot:         [5]6  - [9]9
Swaps:         [5<->5]6<->6  - [8<->9]10<->9
Output array:  [4]4  - [5]6  -  - [7]8  - [8]9  - [9]10
---------------------------------------------
```

Exercise 2
Insert the following sequence of keys into an initially empty hash table. The hash table has a size equal to M=23. Insertions occur character by character using open addressing with double hashing. Use functions h1(k)=kEach character is identified by its index in the English alphabet (i.e., A=1, ..., Z=26). Equal letters are identified by a different subscript (i.e., A and A become A1 and A2).

```
B I T T E R
```

Indicate in which elements are placed the last three letters of the sequence, i.e., T, E, and R, in this order. Please, report your response as a sequence of integer values separated by one single space. No other symbols must be included in the response. This is an example of the response format: 3 4 11

```
### hash-table
Character keys
Hash table size = 23
Number of keys  =  6
Double hashing.
Key:  B I T T E R
Key=B k=002 Final=02
Key=I k=009 Final=09
Key=T k=020 Final=20
Key=T k=020 Coll.=20 Final=18
Key=E k=005 Final=05
Key=R k=018 Coll.=18 Final=14
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22
-  -  B  -  -  E  -  -  -  I  -  -  -  -  R  -  -  -  T  -  T  -  -
```

Exercise 3
Define the data structure heap as adopted in computer science and specify in which problems it is usually used. Report (in C code) the implementation of the function modifying an existing key into a heap. Suppose the heap stores integer values. Report the C data structure type used to define the heap.

Exercise 4
In an activity set, the i-th activity is identified by the pair [si, fi), where si is the starting time and fi is the finishing time. The activities are numbered starting from 1. The following is a correct set of activities.

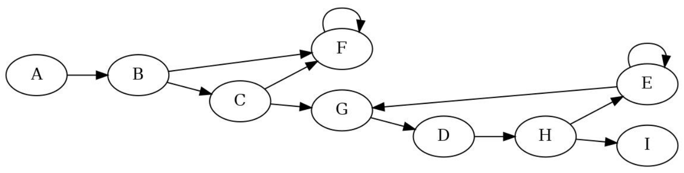
```
 P1  18  21
 P2   5   9
 P3  23  26
 P4   4   7
 P5   2   4
 P6  27  31
 P7  29  33
 P8  28  30
 P9  14  20
P10  30  34
```

Using a greedy algorithm, find the largest subset of mutually compatible activities. Please, report the set of compatible activities in the same order they have been selected, separated by a single space. No other symbols must be included in the response. This is an example of the response: 1 3 2 6 8

```
### Activity selection
Number of activities: 10
 P1  18  21
 P2   5   9
 P3  23  26
 P4   4   7
 P5   2   4
 P6  27  31
 P7  29  33
 P8  28  30
 P9  14  20
P10  30  34
Selected activities:
 P5 [  2,  4)
 P4 [  4,  7)
 P9 [ 14, 20)
 P3 [ 23, 26)
 P8 [ 28, 30)
P10 [ 30, 34)
```

Exercise 5
Visit the following graph in depth-first, starting at node A. Label each edge as tree (T), back (B), forward (F), and cross (C). When necessary, consider nodes and edges in alphabetic order.

```
  A B C D E F G H I
A 0 1 0 0 0 0 0 0 0
B 0 0 1 0 0 1 0 0 0
C 0 0 0 0 0 1 1 0 0
D 0 0 0 0 0 0 0 1 0
E 0 0 0 0 1 0 1 0 0
F 0 0 0 0 0 1 0 0 0
G 0 0 0 1 0 0 0 0 0
H 0 0 0 0 1 0 0 0 1
I 0 0 0 0 0 0 0 0 0
```



Report the label of the edges EG, HI, and BF in this order. Please, indicate the edge type of these 3 arcs with a single letter, i.e., T, F, B, C, separated by single spaces. No other symbols must be included in the response. This is an example of the response: F T B

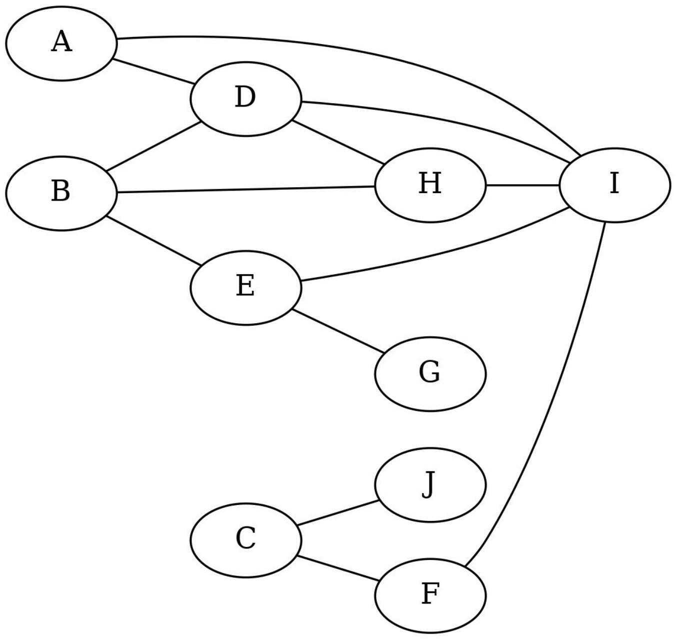
```
Directed matrix ... stated as such.
UN-weighted matrix ...stated as such.
Vertex list: [ 0] A [ 1] B [ 2] C [ 3] D [ 4] E [ 5] F [ 6] G [ 7] H [ 8] I
Adjacency Matrix:
 0 1 0 0 0 0 0 0 0
 0 0 1 0 0 1 0 0 0
 0 0 0 0 0 1 1 0 0
 0 0 0 0 0 0 0 1 0
 0 0 0 0 1 0 1 0 0
 0 0 0 0 0 1 0 0 0
 0 0 0 1 0 0 0 0 0
 0 0 0 0 1 0 0 0 1
 0 0 0 0 0 0 0 0 0

### Graph visit
BFS:
Node=[ 0] A Discovery_time= 0 Predecessor=[-1]
Node=[ 1] B Discovery_time= 1 Predecessor=[ 0]
Node=[ 2] C Discovery_time= 2 Predecessor=[ 1]
Node=[ 3] D Discovery_time= 4 Predecessor=[ 6]
Node=[ 4] E Discovery_time= 6 Predecessor=[ 7]
Node=[ 5] F Discovery_time= 2 Predecessor=[ 1]
Node=[ 6] G Discovery_time= 3 Predecessor=[ 2]
Node=[ 7] H Discovery_time= 5 Predecessor=[ 3]
Node=[ 8] I Discovery_time= 6 Predecessor=[ 7]

DFS:
List of edges:
[ 0] A -> [ 1] B : (T) Tree
[ 1] B -> [ 2] C : (T) Tree
[ 2] C -> [ 5] F : (T) Tree
[ 5] F -> [ 5] F : (B) Back
[ 2] C -> [ 6] G : (T) Tree
[ 6] G -> [ 3] D : (T) Tree
[ 3] D -> [ 7] H : (T) Tree
[ 7] H -> [ 4] E : (T) Tree
[ 4] E -> [ 4] E : (B) Back
[ 4] E -> [ 6] G : (B) Back
[ 7] H -> [ 8] I : (T) Tree
[ 1] B -> [ 5] F : (F) Forward
List of vertices:
[ 0] A: dist_time= 1 endp_time=18 pred=[-1] -
[ 1] B: dist_time= 2 endp_time=17 pred=[ 0] A
[ 2] C: dist_time= 3 endp_time=16 pred=[ 1] B
[ 3] D: dist_time= 7 endp_time=14 pred=[ 6] G
[ 4] E: dist_time= 9 endp_time=10 pred=[ 7] H
[ 5] F: dist_time= 4 endp_time= 5 pred=[ 2] C
[ 6] G: dist_time= 6 endp_time=15 pred=[ 2] C
[ 7] H: dist_time= 8 endp_time=13 pred=[ 3] D
[ 8] I: dist_time=11 endp_time=12 pred=[ 7] H
Reachable nodes from vertex [ 0] A:
[ 0] A - [ 1] B - [ 2] C - [ 3] D - [ 4] E - [ 5] F - [ 6] G - [ 7] H - [ 8] I -
UN-reachable nodes from vertex [ 0] A:
none
```

Exercise 6
Given the following graph find all bridges. If necessary, consider nodes and edges in alphabetical order.

```
  A B C D E F G H I J
A 0 0 0 1 0 0 0 0 1 0
B 0 0 0 1 1 0 0 1 0 0
C 0 0 0 0 0 1 0 0 0 1
D 1 1 0 0 0 0 0 1 1 0
E 0 1 0 0 0 0 1 0 1 0
F 0 0 1 0 0 0 0 0 1 0
G 0 0 0 0 1 0 0 0 0 0
H 0 1 0 1 0 0 0 0 1 0
I 1 0 0 1 1 1 0 1 0 0
J 0 0 1 0 0 0 0 0 0 0
```



Display all bridges. Please, report only the list of edges separated by a single space. The edges must be specified in alphabetic order and within each edge the two vertices must be specified in alphabetic order (i.e., AC and not CA). No other symbols must be included in the response. This is an example of the response: AB AZ BC etc.

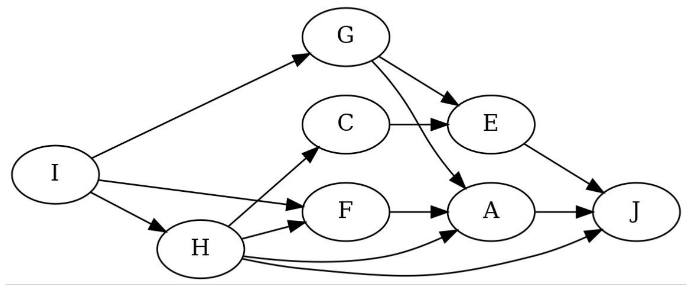
```
UN-directed matrix ... stated as such.
UN-weighted matrix ...stated as such.
Vertex list: [ 0] A [ 1] B [ 2] C [ 3] D [ 4] E [ 5] F [ 6] G [ 7] H [ 8] I [ 9] J
Adjacency Matrix:
 0 0 0 1 0 0 0 0 1 0
 0 0 0 1 1 0 0 1 0 0
 0 0 0 0 0 1 0 0 0 1
 1 1 0 0 0 0 0 1 1 0
 0 1 0 0 0 0 1 0 1 0
 0 0 1 0 0 0 0 0 1 0
 0 0 0 0 1 0 0 0 0 0
 0 1 0 1 0 0 0 0 1 0
 1 0 0 1 1 1 0 1 0 0
 0 0 1 0 0 0 0 0 0 0

### Graph Articulation Points and Bridges
Bridge: Edge [ 4]  E - [ 6]  G
Bridge: Edge [ 2]  C - [ 9]  J
Bridge: Edge [ 5]  F - [ 2]  C
Bridge: Edge [ 8]  I - [ 5]  F
Articulation point: Vertex [ 2]  C
Articulation point: Vertex [ 4]  E
Articulation point: Vertex [ 5]  F
Articulation point: Vertex [ 8]  I
```

Exercise 7

Given the following unweighted DAG, find the topological order of its vertices. Start the DFS visit from vertex A. If necessary, consider nodes in alphabetical order.

```
  A C E F G H I J
A 0 0 0 0 0 0 0 1
C 0 0 1 0 0 0 0 0
E 0 0 0 0 0 0 0 1
F 1 0 0 0 0 0 0 0
G 1 0 1 0 0 0 0 0
H 1 1 0 1 0 0 0 1
I 0 0 0 1 1 1 0 0
J 0 0 0 0 0 0 0 0
```



Report the direct topological order of the vertices. Report vertices on the same line separated by a single space. No other symbols must be included in the response. This is an example of the response format: F A C H B etc.

```
Directed matrix ... stated as such.
UN-weighted matrix ...stated as such.
Vertex list: [ 0] A [ 1] C [ 2] E [ 3] F [ 4] G [ 5] H [ 6] I [ 7] J
Adjacency Matrix:
 0 0 0 0 0 0 0 1
 0 0 1 0 0 0 0 0
 0 0 0 0 0 0 0 1
 1 0 0 0 0 0 0 0
 1 0 1 0 0 0 0 0
 1 1 0 1 0 0 0 1
 0 0 0 1 1 1 0 0
 0 0 0 0 0 0 0 0

### Graph DAG
DFS (discovery and end-processing times):
[ 0] A: dist_time= 1 endp_time= 4
[ 1] C: dist_time= 5 endp_time= 8
[ 2] E: dist_time= 6 endp_time= 7
[ 3] F: dist_time= 9 endp_time=10
[ 4] G: dist_time=11 endp_time=12
[ 5] H: dist_time=13 endp_time=14
[ 6] I: dist_time=15 endp_time=16
[ 7] J: dist_time= 2 endp_time= 3

Topological sort (inverse):
[ 7] [ 0] [ 2] [ 1] [ 3] [ 4] [ 5] [ 6]
  J    A    E    C    F    G    H    I

Topological sort (direct):
[ 6] [ 5] [ 4] [ 3] [ 1] [ 2] [ 0] [ 7]
  I    H    G    F    C    E    A    J

Shortest Paths:
  I    H    G    F    C    E    A    J
infy infy infy infy infy infy    0    1

Longest Paths:
  I    H    G    F    C    E    A    J
infy infy infy infy infy infy    0    1
```

Exercise 8

Analyze the following program and indicate the exact output it generates. lease, report the exact program output with no extra symbols.

```
#include <stdio.h>
#include <string.h>

int f (char *, char *);

int main(void) {
  char *s1 = "first-string";
  char *s2 = "second-string";
  fprintf (stdout, " - %d\n", f (s1, s2));
  return (1);
}
int f (char *s1, char *s2) {
  int i, j, n;
  n = 0;
  for (i=0; i<strlen (s1); i++) {
    for (j=0; j<strlen (s2); j++) {
      if (s1[i]==s2[j]) {
        n++;
        printf ("%c", s1[i]);
        break;
      }
    }
  }
  return (n);
}
```

Exercise 9

The following set of functions implements the heapsort algorithm. Indicates which ones of the following statements are correct. Note that incorrect answers imply a penalty in the final score.

```
#define PARENT(i)   (((i)-1)>>1)
#define LEFT(i)     (((i)<<1)+1)
#define RIGHT(i)    (((i)<<1)+2)
struct heap_s {
  int *v;
  int size;
};
static void swap (heap_t *heap, int i ,int j) {
  int tmp;
  tmp = heap->v[i];
  heap->v[i] = heap->v[j];
  heap->v[j] = tmp;
  return;
}
static void heapify (heap_t *heap, int i) {
  int l, r, tmp;
  l = LEFT(i);
  r = RIGHT(i);
  if ((l<heap->size) && (heap->v[l]<heap->v[i]))
    tmp = l;
  else
    tmp = i;
  if ((r<heap->size) && (heap->v[r]<heap->v[tmp]))
    tmp = r;
  if (tmp != i) {
    swap (heap, i, tmp);
    heapify_min (heap, tmp);
  }
  return;
}
static void build_heap (heap_t *heap) {
  int i;
  for (i=(heap->size>>1)-1; i>=0; i--) {
    heapify (heap, i);
  }
  return;
}
void heapSort (heap_t *heap) {
  int i, size;
  build_heap (heap);
  size = heap->size;
  for (i=heap->size-1; i>=0; i--) {
    swap (heap, 0, i);
    heap->size--;
    heapify (heap, 0);
  }
  heap->size = size;
  return;
}
```

To avoid compilation errors, we must insert the prototypes of all functions on top of the file.

Exercise 10
Analyze the following program and indicate the exact output it generates. Please, report the exact program output with no extra symbols.

```
#include <stdio.h>
void f1 (int);
void f2 (int);
void f3 (int);
void f1 (int n) {
  if (n<=0) {
    return;
  }
  printf ("1");
  f2 (n-2);
  return;
}
void f2 (int n) {
  if (n<=0) {
    return;
  }
  printf ("2");
  f3 (n);
  return;
}
void f3 (int n) {
  if (n<=0) {
    return;
  }
  printf ("3");
  f1 (n+1);
  return;
}
int main () {
  f1(5);
  fprintf (stdout, "\n");
  return 1;
}
```

Exercise 11
An array v stores n real values. Write the function

```
void searchSubArray (int *v, int n, int k);
```

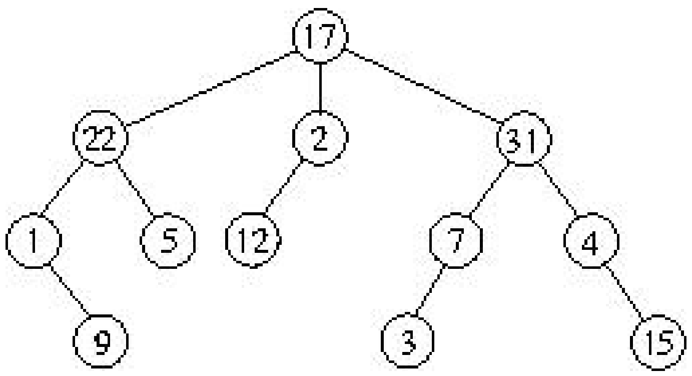
which receives in input the array v of size n, and displays two subarrays of v of k contiguous elements: The one whose sum of the elements is maximum and the one whose difference between the largest and the smallest elements is maximum.
For example, if n=10, k=3, and the array v if the following one 12.5 2.1 3.3 4.1 5.4 6.2 7.9 8.3 -9.9 5.1 the function must display the arrays 12.5 2.1 3.3 (maximum sum, i.e., 12.5+2.1+3.3=17.9) and 8.3 -9.9 5.1 (maximum difference, i.e., 8.3-(-9.9)=18.2).
Write the entire program using standard C libraries but implement all required personal libraries. Modularize the program adequately, and report a brief description of the data structure and the logic adopted in plain English. Unclear or awkward programs, complex or impossible to understand, will be penalized in terms of the final evaluation.

Exercise 12
A n-ary tree stores integer keys. A standard visit would print its key in pre, in, or post-order. Write the function

```
void display (node_t *root);
```

which receives the root of a tree and displays the key of the root, followed by all the keys of the nodes at depth one in alphabetic order, followed by all the keys of the nodes at depth two in alphabetic order, followed by all the keys of the nodes at depth three in alphabetic order, etc.
For example, if the function receives the pointer to the root of the following tree

JPEG
is should display the nodes with the following order: 17 - 2 22 31 - 1 4 5 7 12 - 3 9 15
Write the entire program using standard C libraries but implement all required personal libraries. Modularize the program adequately, and report a brief description of the data structure and the logic adopted in plain English. Unclear or awkward programs, complex or impossible to understand, will be penalized in terms of the final evaluation.



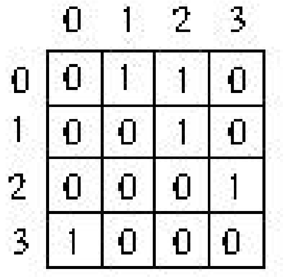
Exercise 13
A directed graph G=(V,E) including n vertices is represented with an adjacency matrix mat.
Write the function

```
void loop (int **mat, int n, int k);
```

which displays all loops in the graph of size k, i.e., all loops including exactly k vertices.
For example, given the following graph
JPEG
the function must print the following sequences:

```
0  2  3
2  3  0
3  0  2
```

Write the entire program using standard C libraries but implement all required personal libraries. Modularize the program adequately, and report a brief description of the data structure and the logic adopted in plain English. Unclear or awkward programs, complex or impossible to understand, will be penalized in terms of the final evaluation.



	0	1	2	3
0	0	1	1	0
1	0	0	1	0
2	0	0	0	1
3	1	0	0	0