**ASSESSMENT AND INTERNAL VERIFICATION FRONT SHEET   
(Grouped Criteria)**

**(Note: This version is to be used for an assignment brief issued to students via Classter)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Course  Title | | BSc. Software Development BSc. Multimedia | | | Lecturer Name & Surname | Andrew Cortis Kassandra Calleja | | |
| Unit Number & Title | | | ITSFT-506-1608 – Data Structures & Algorithms | | | | | |
| Assignment Number, Title / Type | | | 2 – Complex Data Structures & Empirical Analysis / Home | | | | | |
| Date Set | | | 05/05/2023 | Deadline Date | 29/05//2023 | | | |
| Student Name | Andrea Baldacchino | | | ID Number | 349000 | | Class / Group | 6.1A |

|  |  |
| --- | --- |
| Assessment Criteria | Maximum Mark |
| *KU3.1: Show analysis of estimate running times and compare implementation of efficient algorithms with inefficient algorithms.*  *KU4.2: Explain and describe in the context of algorithms, the features correctness, proof and intractability.*  *AA2.3: Produce an algorithm for Graphs or Tree structures. Also compute the best, worst and average case times.*  *AA2.5: Produce an algorithm using the queue data structure to prioritize data. Use a data structure such as a Heap and the Heapsort algorithm.*  ***Total Mark*** | 24 |

|  |
| --- |
| **Notes to Students:** |
| * This assignment brief has been approved and released by the Internal Verifier through Classter. * Assessment marks and feedback by the lecturer will be available online via Classter (<http://mcast.classter.com>) following release by the Internal Verifier * Students submitting their assignment on Moodle/Turnitin will be requested to confirm online the following statements:   **Student’s declaration prior to handing-in of assignment**   * I certify that the work submitted for this assignment is my own and that I have read and understood the respective Plagiarism Policy   **Student’s declaration on assessment special arrangements**   * I certify that adequate support was given to me during the assignment through the Institute and/or the Inclusive Education Unit. * I declare that I refused the special support offered by the Institute. |

# 

**ITSFT-506-1608   
Data Structures & Algorithms**

*First Year BSc. 2022-2023*

Assignment 2

**Complex Data structures & Empirical Analysis**

**Assignment Guidelines**

Read the following instructions carefully before you start the assignment.

If you do not understand any of them, ask your lecturer.

* This is a **HOME** Assignment to be completed by the **DEADLINE SPECIFIED BY LECTURER ON VLE**.
* The assignment consists of **2 Sections and CARRIES 24marks**; all tasks must be attempted.
* Please note that **ALL WORK** must be handed in by the stipulated deadlines.   
  **LATE ASSIGNMENTS WILL NOT BE ACCEPTED.**
* The assignment sheet and assignment coversheet should be fully completed with all the necessary details. Note that **assignments handed in without the assignment cover sheet are considered as not submitted.**
* Assignments ***must* be handed in as a soft copy uploaded on Moodle** by the stipulated deadline.
* Any **references should be listed and quotes should be paraphrased *properly***. Unless listed and paraphrased properly the assignment will be regarded as plagiarized. **Referencing should be carried out using IEEE Style** Referencing Notation.
* **Copying is strictly prohibited and will be penalized** in line with the College’s disciplinary procedures.

**Section A** *(AA2.3, AA2.5)*  
**14marks**

**Scenario:**

CoDex PLC, a fictitious company, is in the process of recruiting new employees. To facilitate its short listing of interview candidates, it has decided to implement a system whereby a custom priority queue, based on a custom Max-Heap is used to identify the most suitable interview candidates. Interview candidates are awarded several points based on their qualifications and years of experience. The priority queue is expected to enque and deque candidates based on points obtained – candidates with the highest number of points are stored at the top of the queue and dequed fist.

**Using Visual Studio as IDE and C# Console Applications, implement the following:**

**Part 1** *(AA2.3, 7marks)*

**Implement a Max-Heap which stores interview candidates with the highest number of points at the top of the heap.**

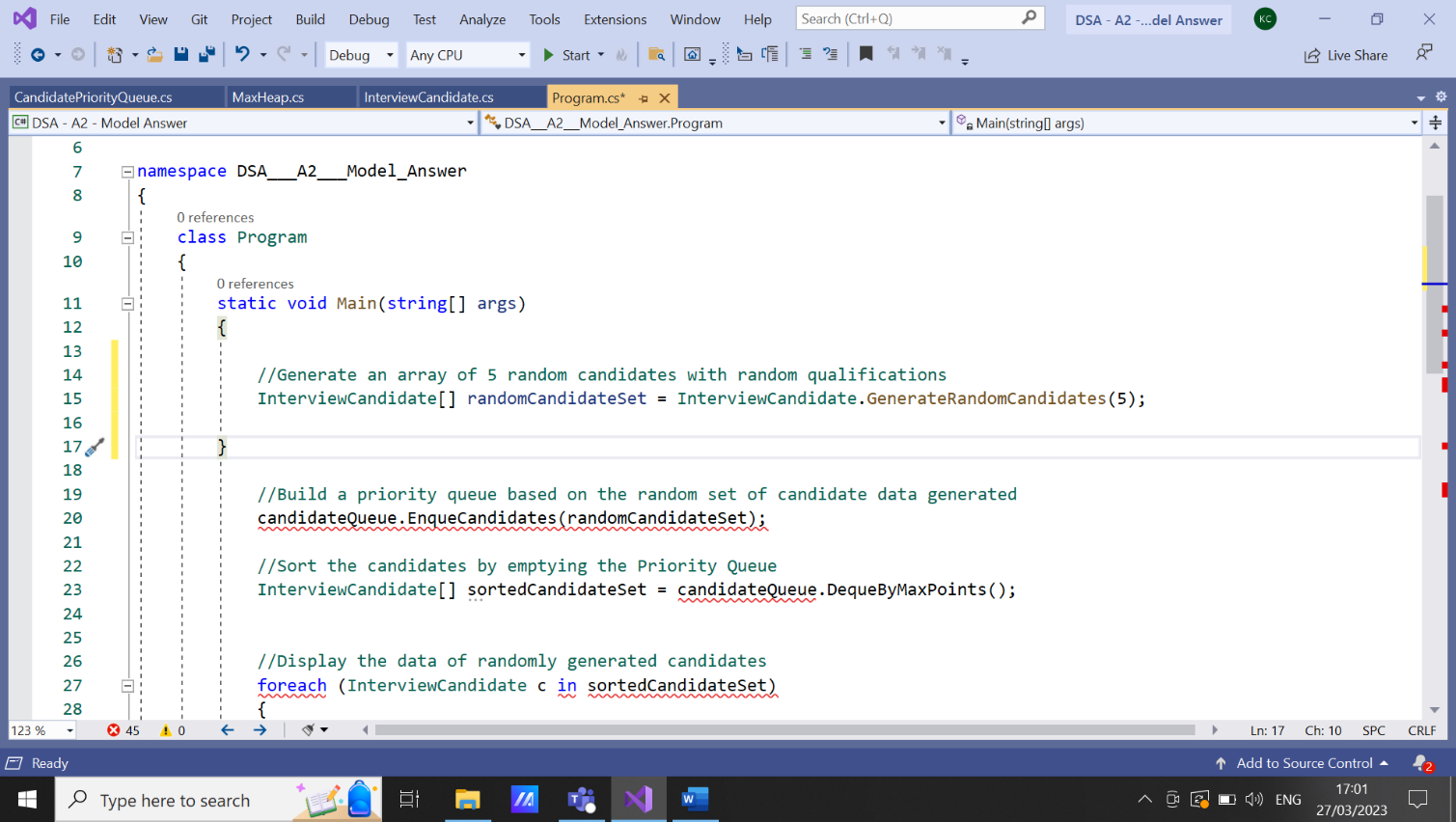
Appendix 1 contains the code for a class Interview Candidate. You are to utilize the code for this class to complete this task; a copy of the .cs file has been made available to you on VLE. You may download the .cs file and import the class into your project – be careful to change the namespace so that it matches that for your project!  
  
The Interview Candidate class contains attributes for the data to be stored for an interview candidate as well as an overloaded constructor and methods to calculate the points achieved by an interview candidate based on his/her qualifications and years of experience.

The class also contains a method :

public static InterviewCandidate[] GenerateRandomCandidates(int numberOfCandidates)

capable of generating an array of the Interview Candidate objects with random data.

You may call this method as follows in Program.cs to obtain an array of random Interview Candidates.



You are to utilize the Interview Candidate class to implement a Max-Heap capable of organizing Interview Candidate objects based on the number of points obtained.

Your Max-Heap implementation must:

1. Use an array of InterviewCandidate objects as the underlying data structure.

**(0.5marks, AA2.3)**

1. The size of the above array must be passed through an overloaded constructor.

**(1 mark, AA2.3)**

1. Use an integer to keep track of the next free location in the heap.

**(0.5marks, AA2.3)**

1. Implement an Insert operation which inserts a new Interview Candidate in the heap, at the correct position, based on the number of points obtained. The Insert method must have the following signature: **(1 mark, AA2.3)**

public void Insert(InterviewCandidate candidate)

and must call within it a HeapifyUp method with the following signature:

**(1.5marks, AA2.3)**

private void HeapifyUp(int index)

1. Implement a Remove operation which removed from the heap, the Interview Candidate with the highest number of points and returns it. The Remove operation must also re-organize the heap so that out of the remaining candidates, the one with the highest number of points is stored at the top of the heap. The Remove method must have the following signature: **(1 mark, AA2.3)**  
     
   public InterviewCandidate Remove()

and must call within it a HeapifyDown method with the following signature:

**(1.5marks, AA2.3)**

private void HeapifyDown(int index)

***You may test your heap implementation using the sample code provided in Appendix 2.***

**Part 2** *(AA2.5, 7marks)*

**Implement a Priority Queue which uses the Max Heap implemented in Part 1 to sort and return Interview Candidates based on number of points obtained, in order highest number of points first.**

Your Priority Queue implementation must:

1. Use the Max-Heap created for Section A Part 1 as the underlying data structure.

**(0.5 marks, AA2.5)**

1. The size of the priority queue / underlying heap must be passed through an   
   overloaded constructor. **(1 mark, AA2.5)**

1. Use an integer to keep track of the number of candidates in the queue. **(0.5 marks, AA2.5)**
2. Implement an EnqueCandidates operation which inserts a set of Interview Candidates in the queue, the interview candidates must be stored based on the number of points obtained – based on qualifications and experience; the candidate with the highest number of points should be the first one in the queue. The EnqueCandidates method must have the following signature:

**(3 marks, AA2.5)**

public void EnqueCandidates(InterviewCandidate[] inputCandidates)

1. Implement a DequeNextCandidate operation which removes the candidate with the highest number of points from the Priority Queue and returns it. The DequeNextCandidate method must have the following signature : **(2 marks, AA2.3)**  
     
   public InterviewCandidate DequeNextCandidate ()

***You may test your Priority Queue implementation using the sample code provided in Appendix 3.***

**Section B** *(KU3.1,KU4.2)*  
**10marks**

**Scenario:**

The Subset Sum Problem aims to determine whether, for a given set of positive integers, a subset of the given set’s elements adds up to a specific sum.

*Example:* If the set-in question is {3, 34, 4, 12, 5, 2} and the required sum is 9.

Do any of the elements in this set add up to 9?

*Yes 🡪 5 and 4 are both within the set and add up to 9.*

Appendix 4 contains the code for a class Subset Sum Problem; it also contains code samples which help you generate random sets of any given size and a random sum value for testing the implementation.

You are to utilize this code to complete tasks in this section; a copy of the .cs file has been made available to you on VLE. You may download the .cs file and import this class into your project – be careful to change the namespace so that it matches that for your project!

**Answer the following questions in relation to the above scenario.**

1. ***The given implementation of the Subset Sum Problem is correct.***
2. What does this mean? **(1mark, KU4.2)**

**This means that it produces the desired output and solves the problem.**

1. Using the following set of values [3,34,4,12,5,2]; write code in Program.cs   
   to prove that the given implementation returns the expected output for   
   a sum = 9 and a sum = 900. **(2marks, KU4.2)**
2. **The given implementation of the Subset Sum Problem is also *intractable*.**
3. What does this mean? **(1mark, KU4.2)**

**[1] That although it does solve the issue, it might take up further resources or time than desired, making it inefficient.**

1. Write code to carry out Empirical Analysis to prove that the Subset Sum Problem is Intractable.  
   - You are to obtain correct and accurate timings for the following 3 input sizes:

5, 50, 500 ***NB: the last value may take some time!***  *Your program is (probably) not stuck, it just needs time. Let it run until it completes.*

**(1.5marks, KU3.1)**

- The timings you obtain must be an average of 20 repetitions. **(0.25 marks, KU3.1)**

* Provide screenshots of the timings you obtained for all input sizes. **(0.25marks,KU3.1)**
* A screenshot of a computer

  Description automatically generated
* A screenshot of a computer

  Description automatically generated
* Based on the timings in your screenshot construct a Log-Log Graph for the Subset Sub Problem. **(1 mark, KU3.1)**
* Comment on the shape of the Log-Log Graph obtained to prove that the algorithm is intractable. **(2 marks, KU3.1)**

***The graph is increasing in efficiency when having a size of 50, but when have a smaller size like 5 it worsened, and became even more inefficient when the size was set to 50, making it an exponential graph.***

**The given implementation of the Subset Sun Problem is also *NP Complete*.**Does this mean that an efficient solution exists for this problem? **(1mark, KU4.2)**

**[2] No, since if there is an efficient solution to an NP complete problem, then there is an efficient solution for all of them.**

Appendix 1

**Code for class Interview Candidate**

class InterviewCandidate

{

//Stores full Name of the Candidate e.g. John Smith

public String CandidateName { get; set; }

//Stores a list of certifications for the candidate where the

//Key is the name of the certification e.g. BSc Software Development

//Value is a number of points awarded to the certification on scale of 1 to 10

public List<KeyValuePair<string, int>> CertificationList { get; set; }

//Stores the number of years of relevant experience

public int yearsOfExperience { get; set;

//Overloaded constructor to instantiate an Interview Candidate object with Data

public InterviewCandidate(string candidateName,

List<KeyValuePair<string, int>>certificationList,

int yearsOfExperience) {

CandidateName = candidateName;

CertificationList = certificationList;

this.yearsOfExperience = yearsOfExperience;

}

/// <summary>

/// Calculates the total number of points to be awarded to a candidate

/// based on certifications and years of experience

/// </summary>

/// <returns>int total number of points obtained by a candidate</returns>

public int CalculateCandidatePoints()

{

int points = 0;

//Add the points awarded to each certification together

foreach(KeyValuePair<string,int> certification in this.CertificationList)

{

points += certification.Value;

}

//Add an additional 2points for every year of experience held by candidate

points += this.yearsOfExperience \* 2;

return points;

}

public static InterviewCandidate[] GenerateRandomCandidates(int numberOfCandidates)

{

//Create an array to hold the required number of interview candidates

InterviewCandidate[] candidates = new InterviewCandidate[numberOfCandidates];

Random dataRandomizer = new Random();

//For each required candidate

for (int c = 0; c < numberOfCandidates; c++)

{

//Generate random string as the candidate name

string fullName = "Candidate Number " + dataRandomizer.Next();

//Generate a random number as the candidate's years of experience

int yearsOfExperience = dataRandomizer.Next(0, 45);

//Generate a random set of qualifications for the candidate

string[] possibleQualifications = {"O Level Certificate",

"A Level certificate",

"BSc.Degree", "MSc. Degree",

"Doctoral Degree",

"Associate Industry Certification",

"Professional Industry Certification"};

//Determine how many qualifications the candidate will have

int numberOfQualifications = dataRandomizer.Next(0, 5);

//Generate the required number of random qualifications and save in a list

List<KeyValuePair<string, int>> qualifications =

new List<KeyValuePair<string, int>>();

for (int i = 0; i < numberOfQualifications; i++)

{

int randomQualificationIndex = dataRandomizer.Next(0, 6);

int randomQualificationValue = dataRandomizer.Next(1, 10);

qualifications.Add( new KeyValuePair<string,int>

(possibleQualifications[randomQualificationIndex],

randomQualificationValue));

}

//Generate a candidate object based on the random data generated

InterviewCandidate candidate = new InterviewCandidate(fullName,

qualifications,

yearsOfExperience);

//Add the candidate to the array to be returned

candidates[c] = candidate;

}

//return the array of candidates

return candidates;

}

}

Appendix 2

**Test Code for MaxHeap.**

static void Main(string[] args)

{

//Create a MaxHeap to store candidates based on points obtained - highest

//amount of points first

MaxHeap candidateHeap = new MaxHeap(15);

//Generate an array of 15 random candidates with random qualifications

InterviewCandidate[] randomCandidateSet =

InterviewCandidate.GenerateRandomCandidates(15);

//Add Each candidate to the MaxHeap

foreach (InterviewCandidate candidate in randomCandidateSet)

{

candidateHeap.Insert(candidate);

}

//Empty the heap, and check that candidates are outputted with the ones

//having the most points first

for(int i = 0; i <15; i++)

{

InterviewCandidate c = candidateHeap.Remove();

Console.Write("\n\nName : {0}\t Experience : {1} years \t

Points {2}\n",c.CandidateName, c.yearsOfExperience,

c.CalculateCandidatePoints());

}

}

Appendix 3

**Test Code for Priority Queue.**

static void Main(string[] args)

{

//Create a Priority Queue to sort candidates based on points obtained - highest

//amount of points first

CandidatePriorityQueue candidateQueue = new CandidatePriorityQueue(15);

//Generate an array of 15 random candidates with random qualifications

InterviewCandidate[] randomCandidateSet =

InterviewCandidate.GenerateRandomCandidates(15);

//Build a priority queue based on the random set of candidate data generated

//These must be stored in a way whereby the candidate with the highest number of

//points is first

candidateQueue.EnqueCandidates(randomCandidateSet);

//Deque the candidates 1 by 1, so that they are displayed in the order highest amount

//of points forst

for (int i = 0; i < 15; i++)

{

InterviewCandidate nextCandidateInQueue = candidateQueue.DequeNextCandidate();

Console.Write("\n\nName : {0}\t Experience : {1} years \t Points {2}\n",

nextCandidateInQueue.CandidateName,

nextCandidateInQueue.yearsOfExperience,

nextCandidateInQueue.CalculateCandidatePoints());

}

Console.ReadKey();

}

Appendix 4

**Class Subset Sum Problem.**

class SubsetSumProblem

{

public static bool sumExists(int[] set, int sum, int n)

{

// Base Cases

if (sum == 0)

return true;

if (n == 0)

return false;

// If last element is greater than sum, then ignore it

if (set[n - 1] > sum)

return sumExists(set, sum, n - 1);

//else, check if sum can be obtained by any of the following

//(a) including the last element

//(b) excluding the last element

return sumExists(set, sum, n - 1) || sumExists(set, sum - set[n - 1], n - 1);

}

}

**Code for Generating random sets of a given size and random sum values.**

static Random rand = new Random();

static void Main(string[] args)

{

//Determine the input size i.e. n

int inputSize = 10;

//Check if a random sum value exits in the random set

IsSumInSubset(inputSize);

Console.ReadKey();

}

***Continues on next page….***

public static bool IsSumInSubset(int sizeOfSet)

{

//Create a set of unique random integers of size n

int[] set = new int[sizeOfSet];

set = Enumerable.Range(0, int.MaxValue).Take(sizeOfSet).ToArray();

//Generate a random sum value

int randomSumValue = rand.Next(0, sizeOfSet);

//Display the set of numbers as a comma-separated string

Console.WriteLine("Set:\n"+String.Join(",", set)+"\n");

//Check if the sum randomValue exists in set

bool sumExits = SubsetSumProblem.sumExists(set, randomSumValue, set.Length);

//Check if the required sum exits

if (sumExits == true)

{

Console.WriteLine("Found a subset of elements which add up to {0}\n\n",

randomSumValue);

return true;

}

else

{

Console.WriteLine("No subset of elements which add up to {0} found\n",

randomSumValue);

return false;

}

}

[1] Paul E. Black, "intractable", in [Dictionary of Algorithms and Data Structures](https://www.nist.gov/dads/) [online], Paul E. Black, ed. 17 December 2004. Available from: <https://www.nist.gov/dads/HTML/intractable.html>

[2] “What is an NP-complete in computer science?,” Stack Overflow, https://stackoverflow.com/questions/210829/what-is-an-np-complete-in-computer-science (accessed May 23, 2023).

End of Assignment