Algorithms & Data Structures
Exercise Sheets

Weeks 3-4: Elementary data structures and their processing

Exercises

(1) (Week 3) Implement a Stack using an ordinary array. The stack should be initialized with a size parameter that defines the initial size of the stack. The default constructor should create a stack with 100 elements as the initial size. When `push(...)` is called and the stack is full, a new array should be allocated that have the double size, the content of the old array copied to the new one and the old array deleted. Analyze the running time of the stack operations `push` and `pop` in terms of the N elements it stores.

(2) (Week 3) Add a `transpose()` member function to the `Matrix` class that replaces a matrix with its transpose. Try to do this without the use of a temporary matrix. What is the resulting worst-case time complexity in Θ notation (i.e. the lowest upper-bound)?

(3) (Week 3) Write a function that takes a *sorted* N by N matrix already stored in memory and decides if a number x is in the matrix with worst-case complexity $O(N)$ (linear in **one** of the dimensions). Assume that each individual row is increasing from left to right and each individual column is increasing from top to bottom. Justify the achieved complexity.

(4) (Week 3) Write an implementation of `Queue` using `Stack`(s). The implementation must be a template and use the Stack ADT presented in the course. There is no requirement that the implementation is efficient, but you cannot use the internal structure of the stack, nor an iterator for the stack.

(5) (Week 4) Let T by a hash-table of size 7 with the hash function $h(x) = x \; mod \; 7$. Write down the entries of T after the keys 5, 28, 19, 15, 20, 33, 12, 17, 33 and 10 have been inserted using

   (a) chaining

   (b) linear probing

   (c) quadratic probing

   What is the load-factor(λ) in the three cases

(6) (Week 4) Implement a `Set` using either a `Queue`, `List` or `Stack` ADT as presented in the course. There is no requirement that the implementation is efficient, but you cannot use the internal structure of the ADT, nor an iterator for the ADT

(7) (Week 4) Implement a `Dictionary` using an STL vector. The element of the vector (for the implementation) must be an STL pair representing the (key, value). You can find an example of how to use pair here: `https://www.geeksforgeeks.org/pair-in-cpp-stl/`

(8) (Week 4) An old exam question

Nedenstående tabel er en hashtabel, hvor der anvendes quadratic probing i tilfælde af kollisioner (collision resolution). Hash funktionen er $h(x) = x \bmod 11$. Følgende værdier er indsat: 22, 5, 16 og 27 (i denne rækkefølge):

Index	Value
0	22
1	
2	
3	
4	
5	5
6	16
7	
8	
9	27
10	

Positionerne (indexes) 1, 2, 3, 4, 7, 8 og 10 er ubrugte. Verificer at hashtabellen er korrekt og ret eventuelle fejl. Vis tabellens udseende efter at elementerne 1, 12, din alder samt dit studienummer er indsat og begrund hvorfor det indsættes hvor det indsættes samt forklar hvad det derudover sker. Noter i besvarelsen hvad din alder og studienummer er.