3805ICT Advanced Algorithms – Assignment 1

Chris Vuong – s5193954

**Question 4:** Design and implement in C++ a data structure for storing unordered lists

of integers that:

* Can store integers in the range 0 .. n where n is some upper bound.
* Duplicate integers are not allowed in the list.
* Is O(1) for add, delete, test for being in the list, and iterating through the list.
* Is O(k) (where k is the number of integers in the list) for clearing the list.

**Idea:** The idea of this data structure is to record whether the number is in the list or not. Therefore, the data structure will include a pointer list and an array of n Boolean values which store if the number is in or out of the list.

**Pseudo-code:**

|  |
| --- |
| Struct unordered\_list {  Int \*data;  Int \*unorder[]; // this array will keep the list unorder  }  Function generate (int n): => unorder\_list  unorder\_list ul;  ul.data = new bool[n];  loop i = 0 to n:  ul.data[i] = -1; // pre-assigned the empty array  ul.unorder = new \*int[];  Return ul;  Function add(unorder\_list ul, int i): => unorder\_list  If ul.data[i]> -1:  Return ul;  ul.data[i] = ul.unorder.size; // add the location of the number should be  ul.unorder.push\_back( &ul.data[i] ); //put the address of the data  Function delete(unorder\_list ul, int i): => unorder\_list  ul.data[i] = -1;  Return ul;  Function test(unorder\_list ul, int i): => bool  Return ui.data[i] > -1;  Function printList(unorder\_list ul): => void  Loop from i=0 to ul.unorder.size:  If \*ul.unorder[i] > -1 and \*ul.unorder[i] == i:  print \*ul.unorder[i]  Function clear(unorder\_list ul):=> void  Loop from i= 0 to ul.unorder.size:  Int n = ui.unorder[i] - list.data; // substract 2 pointers to get the value  Delete(ul , n)  Ul.unorder = new \*int[];  Function Iteration(unorder\_list ul) => int[]  Int[] result = new int[];  Loop from i= 0 to ul.unorder.size:  Result.push\_back( ui.unorder[i] - list.data ); // substract 2 pointers to get the value  Return result; |

**Performance:**

|  |  |  |
| --- | --- | --- |
| Generation | O(n) | For pre-assigning all the value in data to -1 |
| Addition | O(1) | Put the value data[i] as the index of the unorder list |
| Deletion | O(1) | Changing the value in data[i] to -1 |
| Test | O(1) | Checking if the value in data[i] > -1 |
| Iteration | O(T) | Giving the output of an array of the unorder list. This will run the T time as T is the total of the number in the list (k) and the number has been deleted. |
| Clearance | O(T) | It will clear all the value in the unorder list in T time. T >= k where k is the number of exist item in the list. |

**Disadvantage**: This data structure will cost a large amount of storage.

For the clearing function, it is cannot run in the k time if there was at least an element deleted.

OUTPUT from c++ :

|  |
| --- |
| Please enter the size n = 20  []  adding 5.  adding 3.  [5, 3]  adding 3.  Error :Duplicate integers are not allowed in the list.  adding 100.  Error :Can only store integers in the range 0 .. n  adding 6.  [5, 3, 6]  testing 3 in the list.  1  deleting 3.  [5, 6]  adding 3:  [5, 6, 3]  testing 4 in the list.  0  clearing the list.  [] |