Assignment9

Samundra karki

May 2019

1(a)

In the sequence <3, 10, 2, 4> we can apply the h_1(k) = k mod 3 and h_2(k)=7k mod 8. So , if key = 3 we have h_1(k) = 3 and h_2(k) = 5.
Here, since there is no collsion and current index 3 is available so, the slot 3 is given to key = 3.
If key = 10, we have h_1(k) = 0 and h_2(k) = 6.
Here, since there is no collision and current index 0 is available so, the slot 0 is given to the key = 10.
If key = 2, we have h_1(k) = 2 and h_2(k) = 6.
Here, since there is stil is no collison and current index 2 is available so, the slot 2 is given to the key = 2.
If key = 4, we have h_1(k) = 4 and h_2(k) = 4.
Here, since position 4 is still available so the slot 4 is given t the key = 4.
Position in order we get: 3, 0, 2, 4.

1(b)

You can find the HashTable.java file in this zip file. I have used division method in hashing.
Why division method?
The goal of hash function is to distribute the keys as uniformly as possible.Here number of the element in array is 200 and nearset prime number to 200/3 is 67. So n = 200 and m = 67.
This increses the uniformity of the hash table created by this hash function.

2(a)

Activity-selection problem : Choosing the problems such that the person can only start the job if it's next acitivity starting time is greater than or equal to the finsihing time of its current acitivity.
This statement will create a flaw in Greedy Agorithm.
Lets take a example that would make us clear about the arising problem.

a_1 = 1 - 10.
a_2 = 9 - 11.
a_2 = 11 - 15.
Here, we are not able to choose the activity with minimum time span because this occurs in the activity two. Due to the choice of the a_1 as our intial optimal solution, we could not choose a_2 due to the fact that 9 ¡ 10.

2(b)

Gist of the greedy algorithm that selectes activity with latest starting time.
Taking a example :
a_1 = 1 - 5.
a_2 = 9 - 15.
a_3 = 5 - 11.
a_4 = 20 - 23.
a_5 = 4 - 8.
a_6 = 3 - 19.
Here, we are not allowed to sort the problem which would cost us minimum of O(nlgn) time complexity by comparision sort. First we should find the latest starting point and make it null and then search next greatest. If the second greatest is disjoint with the current greatest then this would be added to optimal solution otherwise not.
Activity class struture:
class Activity{
startingTime;
finsihingTime;
}

---

findMax(Activity)
tempMax = Activity[1].startingTime
index = 1;
for i = 2 to Activity.size **do**
  if Activity[i].startingTime > tempMax **then**
    index = i;
  end if
end for
return index

---

isDisjoint(staringTime, finishingTime)
return finshingTime <= startingTime

ActivitySelection(Activity)
create new array for result R

```
for i = 1 to Activity.size do
  maxIndex = findMax(Activity)
  if (R is empty) then
    add Activity[maxIndex] to R
    make the current index of the null.
  end if
  if isDisjoint(Activity[maxIndex].finishingTime, R[R.size].startingTime) then

    add the current index i,e Activity[maxIndex] to the R
    make the current index of the null.
  end if
end for
return  R
```

Here **isDisjoint** returns the boolean value if the element present in result array's startingTime is greater than the current maximum or latest staring Activity index passed. Here we make current max to null to know another maxElement when called again.