CSC 2221: Algorithms

Greedy Algorithms

Supta Richard Philip
Lecturer

Department of CSE
American International University(AIUB).

AIUB, October 2022

Table of Contents

Table of Contents

Conventional Algorithms Design Techniques

- **Divide and Conquer:** The Divide and Conquer strategy involves dividing the problem into sub-problem, recursively solving them, and then recombining them for the final answer. Example, Merge sort, Quicksort.
- **Greedy Method**: In the greedy method, at each step, a decision is made to choose the local optimum, without thinking about the future consequences. Example, Fractional Knapsack, Activity Selection.

Conventional Algorithms Design Techniques

- **Dynamic Programming:** The approach of Dynamic programming is similar to divide and conquer. The difference is that whenever we have recursive function calls with the same result, instead of calling them again we try to store the result in a data structure in the form of a table and retrieve the results from the table. Thus, the overall time complexity is reduced.
  Example: 0-1 Knapsack, subset-sum problem.
- **Backtracking:** This is useful to solve combinatorial problems that have a single unique solution.
  Example: N-queen problem, maize problem.
- **Branch and Bound:** This is useful to solve combinatorial optimization problem that have multiple solutions.
  Example: Travelling salesman problem.

Table of Contents

What is Greedy Algorithm?

- A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment.
- Greedy algorithms are often used to solve optimal problems (find best solutions of the problem according to a particular criteria).
- Greedy algorithms implement optimal local selections and hope that those selections will lead to an optimal global solution for the problem.
- Drawback of Greedy Approach, Some times the current best result will not bring the overall optimal result.
- The algorithm never reverses the earlier decision even if the choice is wrong. It works in a top-down approach.

Properties of Greedy Algorithm

- **Greedy Choice Property:** If an optimal solution to the problem can be found by choosing the best choice at each step without reconsidering the previous steps once chosen, the problem can be solved using a greedy approach. This property is called greedy choice property.
- **Optimal Substructure:** If the optimal overall solution to the problem corresponds to the optimal solution to its subproblems, then the problem can be solved using a greedy approach. This property is called optimal substructure.

Table of Contents

Coin change problem

- Given currency denominations: 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000 devise a method to pay amount to customer using fewest number of coins/ notes.
- Amount: 30
  Solutions : 3 X 10 (3 coins)
  6 X 5 (6 coins)
  1 X 20 + 5 X 2 (3 coins)
  1 X 20 + 1 X 10 (2 coins)
- The last solution is the optimal one as it gives us a change of amount only with 2 coins, where as all other solutions provide it in more than two coins.

Counter Example

- For coin denominations 1, 5, 10, 20, 25, it need not give optimal solution. Counter example: amount=40.
  Greedy algorithm gives, 1×25, 1×10, 1×5.
  Optimal is 2×20.
- It is true that a greedy algorithm can not always solve an optimization problem.

Coin change problem Algorithm/Code

```
void findMin(int total){
int deno[10]={1,2,5,10,20,50,100,200,500,1000};
vector<int> ans;
for(int i=9;i>=0;i--){
while(total>=deno[i]){
total = total - deno[i];
ans.push_back(deno[i]);
}
}
for(int i=0;i<ans.size();i++){
cout<<ans[i]<<" ";
}
}
```

Table of Contents

Fractional Knapsack problem

- Suppose the items are 1, 2, ..., n.
- Weight of item i is Wi and value/profit/Cost is Ci.
- sort the items in non-increasing(descending order) order of Ci/Wi.
- Start adding the item with the maximum Cost / Weight
- Add the whole item, if the current weight is less than the capacity, else, add a portion of the item to the knapsack.
- Stop, when all the items have been considered and the total weight becomes equal to the weight of the given knapsack.

Fractional Knapsack problem

- For the given set of items and knapsack capacity = 12 kg, find the optimal solution using the fractional knapsack algorithm.
- Item: 1 2 3 4
- Profit/Value/Cost(in Taka) 300 450 500 600
- Weight(in kg): 3 5 8 10
- Ratio (Pi/Wi): 100 90 62.5 60
- Max Profit: 1000

Questions

- Given a value of N Tk. and an infinite supply of each of the denominations 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000 valued coins or notes, The task is to find the minimum number of coins and notes needed to make the change 380 Tk.? Show the simulation or write the algorithm.
- For the given set of items and knapsack capacity = 60 kg, find the optimal solution using the fractional knapsack algorithm.
  Item: 1 2 3 4
  Profit: 280 100 120 120
  Weight: 40 10 20 24
  Ratio (Pi/Wi): 7 10 6 5