CMPT 405 Assignment I: Greedy algorithm

1. The *fractional knapsack problem* is defined in this way: given materials of different values per unit volume and maximum amounts, find the most valuable mix of materials which fit in a knapsack of fixed volume. Note now we may take pieces (fractions) of materials. More formally:

   Given a knapsack of capacity $c > 0$ and N items. Each item has value $v_i > 0$ and weight $w_i > 0$. Find the selection o f items ($\delta_i \in [0, 1]$) that fit ($\sum_{i=1}^{N} (\delta_i \cdot w_i) \leq c$) and $\sum_{i=1}^{N} (\delta_i \cdot v_i)$ is maximized.

   Propose a greedy algorithm to solve this problem. Show why your algorithm is optimal.

   Answer: Let W and V denote the total weights and values of all items in the bag. We design the following algorithm (*GREEDY*) to solve the fractional knapsack problem:

```
procedure GREEDY {
  W = 0;
  V = 0;
  sort all items according to v_i/w_i by decreasing order;
  for i from1 to N {
    if (W + w_i ≥ c) break;
    else{
      δ_i = 1;
      W += w_i;
      V += v_i;
    }
  }

  δ_i = (c − W)/w_i;
  V += δ_i · v_i;

  for j from i + 1 to N
    δ_j = 0;
}
```

   In the following we show the algorithm *GREEDY* is optimal.

   Proof (Sketch) Let k be the number of different items in the optimal solution. We prove GREEDY is optimal by induction on k. If $k = 1$, it has to be the one with maximum value/weight rate, in this case GREEDY is optimal. We assume when $k \geq 1$, GREEDY always outputs the optimal solution. Consider the case which the optimal solution contains $k+1$ different items. Notice the optimal solution has to include the whole item with maximum value/weight rate, say item i. We are able to find it by the first step of GREEDY. Now we remove item i from the item set, and reduce the total weight of the knapsack from c to $c - w_i$. After then, we get a subproblem where the optimal solution only includes k items. By induction hypothesis, The rest of GREEDY can find the optimal solution for the subproblem. Thus GREEDY is guaranteed to find the optimal solution for the case when the optimal solution with $k + 1$ different items. Hereby GREEDY is optimal. ∎

2. The *0/1 knapsack problem* is similar to the fractional knapsack problem, only now we are not allowed to have pieces (or fractions) of materials. The following is the formal definition of *0/1 knapsack problem*:

   Given a knapsack of capacity $c > 0$ and N items. Each item has value $v_i > 0$ and weight $w_i > 0$. Find the selection of items ($\delta_i = 1$ if selected, 0 if not) that fit ($\sum_{i=1}^{N} (\delta_i \cdot w_i) \leq c$) and $\sum_{i=1}^{N} (\delta_i \cdot v_i)$ is maximized.

   It is well known that the 0/1 knapsack algorithm is NP-complete. Thus there (seems) does not exist a polynomial algorithm which can give optimal solution. But we may slightly change the greedy algorithm in Q1 (named $GREEDY$) to get a 2-approximation algorithm for 0/1 knapsack problem. Note now we restrict $GREEDY$ to only take integral objects. Let v_{max} be the maximum value of all items, V_{GREEDY} be the result of the new $GREEDY$ algorithm.

   Step1: Apply the restricted $GREEDY$ algorithm;
   Step2: Return $\max\{V_{GREEDY}, v_{max}\}$;

   Prove the optimal solution is at most two times of the result of the above algorithm for all feasible inputs.

   Proof Let j be the first index at which GREEDY stops. Define $\overline{v_j} = \sum_{i=1}^{j-1} v_i$, $\overline{w_j} = \sum_{i=1}^{j-1} w_i$. Let OPT be the optimal solution. Because $\overline{w_j} + w_j > c$, and all the items are sorted according to their value/weight rates, we may conclude that $OPT < \overline{v_j} + v_j$. Also we note that $V_{GREEDY} = \overline{v_j}$, and $v_{max} \geq v_j$, thus $OPT < \overline{v_j} + v_j \leq 2 \cdot \max\{V_{GREEDY}, v_{max}\}$. ∎

3. A k-coloring of a graph $G = (V, E)$ is a mapping $f : V \to 1, 2, \ldots, k$ such that adjacent vertices are mapped out different colors (one may think of numbers $1, \ldots, k$ as "colors", i.e., no two neighbors in the graph G receive the same color).

   Prove the following GREEDY algorithm colors the given graph G with at most $\Delta(G) + 1$ colors, where $\Delta(G)$ denotes the maximum degree (number of adjacent vertices) of any node $v \in V$.

   For $i = 1, \ldots, n$ do
   Color vertex v_i using the smallest available color in $1, \ldots, \Delta(G) + 1$.

   Proof By contradiction. Suppose at one time we have to use the $(\Delta(G) + 2)_{th}$ color to color a vertex v. Then we have to use every color in $1, 2, \ldots, \Delta(G) + 1$ at least once to color all the neighbors of v. But v only has at most $\Delta(G)$ neighbors. We get a contradiction. ∎

4. We need to make change for n cents using the least coins as possible.

   (a) Describe a greedy algorithm to make change consisting of quarters (25), dimes(10), nickels (5), and pennies (1). Proof your algorithm is optimal.

   (b) Suppose we only have quarters, dimes and pennies, is the greedy algorithm still optimal? Explain why.

   Answer:

   (a) The following algorithm makes change of n cents. Suppose we have m different coins. Let c_i ($i \in [1, m]$) be the denomination of the i_{th} coins. Let k_i be the number of the i_{th} coins.

```
procedure make-change{
  for i from 1 to m{
    k_i = n/c_i;
    n- = (n/c_i) · c_i;
  }
}
```

The greedy make change algorithm is optimal for the (25, 10, 5, 1) denomination. Now we are going to show it.

Proof (sketch) First we observe that in the optimal solutions the number of dimes can only be $0, 1, 2$, the number of nickels can only be $0, 1$, the number of pennies can only be $0, 1, 2, 3, 4$. Besides, it is not possible to have two dimes and one nickels at the same time. (these observations can all be easily show if the reverse happens, then the solution can not be optimal, we omit here). Thus all the dimes, nickels and pennies in the optimal solutions can at most form a 24. Hence, our greedy make change algorithm finds correct number of quarters. Keep using the same proving technique (can you think how?), we can show the numbers of dimes, nickels and pennies the greedy make change algorithm finds are all optimal too. ∎

(b) Consider the example when $n = 34$. the greedy algorithm outputs 1 quarters and 9 pennies. But the optimal solution is 3 dimes and 4 pennies.