Greedy Algorithms

Fractional knapsack

Coin changing
Fractional knapsack

- **Input:**
  - $n$ items with weight $w_1, w_2, \ldots, w_n$ and values $v_1, v_2, \ldots, v_n$
  - A knapsack with capacity $W$ (total weight of items at most $W$)
  - The items are divisible: can put a fraction of an item into knapsack

- **Output:** maximize $p_1v_1 + p_2v_2 + \ldots + p_nv_n$  
  Constraint: $p_1w_1 + p_2w_2 + \ldots + p_nw_n \leq W$
  - Where $p_i$ are the fraction of item $i$
Fractional knapsack

- **Greedy Algorithm 1:**
  - Sort items in decreasing order of value (take most valuable first)
  - $120 + 100 = 220$ Not optimal
Fractional knapsack

- **Greedy Algorithm 2:**
  - Sort item in increasing order of weight (take lightest item first)
  - $60 + 100 + (20/30) \times 120 = 240$
  - Looks optimal for this example.
  - How about this example: values = [20, 50, 100] amd weights=[10, 20, 100] and $W=10$
Fractional knapsack

- Greedy Algorithm 3:
  - Sort items in decreasing order of \(\frac{v_i}{w_i}\)
Fractional knapsack: greedy algorithm

- Pseudocode
  ```python
def Knapsack(W, v[1...n], w[1...n]):
    S = {}
    for i = 1 to n:
        if w_i < W:
            Put item i into S
            W = W-w_i
            V = V + V_i
        else:
            #put fraction of item i into S
            V = V + V_i / w_i
            break
    return V
  ```

- Runtime Analysis
  - $O(n \log n)$
The greedy algorithm gives an optimal solution to the fractional knapsack problem.

Proof:

- \( X \): Greedy solution = \( \{x_1, x_2, \ldots, x_n\} \)
- \( Y \): Optimal solution = \( \{y_1, y_2, \ldots, y_n\} \)
- We prove that \( X = Y \)
  - We prove that if the optimal solution is different from greedy algorithm, then there will be a solution that is better than optimal solution (has more value than optimal solution) which is a contradiction.
- Suppose \( X \) is not equal to \( Y \)
- Let \( j \): the smallest integer \( j \) such that \( x_j \neq y_j \)
Fractional knapsack: greedy algorithm 3: Proof of Optimality

Greedy: $X = x_1 \ x_2 \ \ldots \ x_{j-1} \ x_j \ \ldots \ x_k \ \ldots \ x_n$

Optimal: $Y = y_1 \ y_2 \ \ldots \ y_{j-1} \ y_j \ \ldots \ y_k \ \ldots \ y_n$

$Y' = y_1 \ y_2 \ \ldots \ y_{j-1} \ y'_j \ \ldots \ y'_k \ \ldots \ y_n$

- $x_j > y_j$ because of the way algorithm works
- Must exist $k > j$ such that $y_k > 0$ and $y_k > x_k$. why?
- Exchange weight $\varepsilon$ of item $k$ for equal weight of item $j$ in an optimal solution

$$y'_j = y_j + \frac{\varepsilon}{w_j} \quad y'_k = y_k + \frac{\varepsilon}{w_k}$$

- The optimal value will be increased by

$$\left(\frac{\varepsilon}{w_j}\right)v_j - \left(\frac{\varepsilon}{w_k}\right)v_k = \varepsilon\left(\frac{v_j}{w_j} - \frac{v_k}{w_k}\right)$$

- Contradiction: $Y$ was optimal
Coin Changing
Coin Changing

**Input:** A set of coin denominations $d_1, d_2, \ldots, d_n$ and a positive integer $M$

**Output:** finding $A = [a_1, \ldots, a_n]$ such that $M = \sum_{i=1}^{n} d_i a_i$ and $\sum_{i=1}^{n} a_i$ is minimized.

- $a_i$ denotes the number of coins of denomination $d_i$ that are used
- The total value of all the chosen coins must be exactly equal to $M$.
  - We want to minimize the number of coins used
1. CASHIERS-ALGORITHM (M, d₁, d₂, ..., dₙ)
2. SORT n coin denominations so that 0 < d₁ < d₂ < ... < dₙ
3. S = [0, ..., n] #S[i] stores number of coins of type dᵢ
4. for i = n to 1
5.  S[i] = floor (M / dᵢ)
6.  M = M - S[i] * dᵢ
7.  if M>0
8.     return “no solution.”
9.  return S
Observations: Properties of optimal solution

Properties of optimal solution for this coin denominations: \{ 1, 5, 10, 25, 100 \}

1. Number of 1 cent coins $\leq 4$:
   
   Replace 5 of them by one 5 cents coin

2. Number of 5 cents coin $\leq 1$:
   
   Replace two of them by 10 cents coin

3. Number of 25 cents coin $\leq 3$:
   
   Replace four of them by a loonie

4. Number of 5 cents coins + number of 10 cents coins $\leq 2$
   
   - Replace three 10 cents and zero 5 cents with one 25 cents and one 5 cents
   - Replace two 10 cents and one 5 cents with one 25 cents
Proof of optimality for US. coin denomination

- **Theorem.** Cashier’s algorithm is optimal for U.S. coins \{ 1, 5, 10, 25, 100 \}
- **Proof:** by induction on amount \( M \)
  - **Base case:** \( M = 1, 2, 3, 4 \)
  - **Induction hypothesis:** the greedy algorithm produces the optimal solution for values less than \( M \)
  - **Induction step:** having the induction hypothesis, show that the algorithm produces the optimal solution for \( M \)
    - The greedy algorithm chooses \( d_k \) such that \( d_k \leq M < d_{k+1} \)
    - Any optimal solution must choose \( d_k \)
      - if not, it needs enough coins of type \( d_1, \ldots, d_{k-1} \) to add up to \( M \)
      - However, no optimal solution can do this
    - Optimal and greedy choose the same for \( M \). Now, we need to show that the algorithm produces optimal solution for \( M - d_k \) cents, which is true by induction
Proof of optimality for US. coin denomination

- Suppose: $5 \leq M < 10$
  - Suppose there is no 5 cents in the optimal solution.
  - The optimal solution contains only of 1 cent → contradiction
    - Number of 1 cent coins is at most 4
  - → There is no combination of lower denomination (1 cent coins) that add up to M and still is optimal
  - Therefore, the greedy solution contains at least one 5 cents coin. By induction the greedy solution is optimal for M-5. → The greedy solution is optimal

Properties of optimal solution for this coin denominations: { 1, 5, 10, 25, 100 }

1. Number of 1 cent coins ≤ 4
2. Number of 5 cents coin ≤ 1
3. Number of 25 cents coin ≤ 3
4. Number of 5 cents coins + number of 10 cents coins ≤ 2
Proof of optimality for US. coin denomination

- Suppose: $10 \leq M < 25$
  - Suppose there is no 10 cents in the optimal solution.
  - The optimal solution contains only of 1 cent and 5 cents coins
  - In an optimal solution, we can get at most 9 using 1 and 5 cents coins
  - → There is no combination of lower denomination (1 and 5 cents coins) that add up to M and still is optimal
  - Therefore, the greedy solution contains at least one 10 cents coin. By induction the greedy solution is optimal for $M-10$. → The greedy solution is optimal

Properties of optimal solution for this coin denominations: { 1, 5, 10, 25, 100 }

1. Number of 1 cent coins $\leq 4$
2. Number of 5 cents coin $\leq 1$
3. Number of 25 cents coin $\leq 3$
4. Number of 5 cents coins + number of 10 cents coins $\leq 2$
Proof of optimality for US. coin denomination

- Suppose: $25 \leq M < 100$
  - Suppose there is no 25 cents in the optimal solution.
  - The optimal solution contains only of 1, 5, and 10 cents coins
  - We can change at most 24 using (10, 5, 1) and still have an optimal solution
    - 2 ten cents and 4 one cent
    - Example: To construct 26 using only (10, 5, 1) we get $10 + 10 + 5 + 1$
      - It contradicts the last observation: Number of 5 cents coins + number of 10 cents coins should be at most 2 which is not in the above example
  - Therefore, the greedy solution contains at least one 25 cents coin. By induction the greedy solution is optimal for $M-25$. → The greedy solution is optimal

Properties of optimal solution for this coin denominations: \{ 1, 5, 10, 25, 100 \}

1. Number of 1 cent coins \leq 4
2. Number of 5 cents coin \leq 1
3. Number of 25 cents coin \leq 3
4. Number of 5 cents coins + number of 10 cents coins \leq 2
Proof of optimality for US. coin denomination

- **Suppose**: $100 \leq M < 200$
  - There is no combination of lower denomination (25, 10, 5, 1) that add up to $M$ and still is optimal
  - If there are no loonies in optimal solution, the optimal solution can have at most 4 (one cent) and 2 (ten cents) and 3 (twenty five cents) which add up to 99
    - → cannot make $M$ → Optimal (and the greedy) contains a loonie.
    - By inductive hypothesis, greedy is optimal for $M-100$. So greedy is optimal for $M$

- **Exercise**: Suppose: $200 \leq M$

Properties of optimal solution for this coin denominations: { 1, 5, 10, 25, 100 }

1. Number of 1 cent coins $\leq 4$
2. Number of 5 cents coin $\leq 1$
3. Number of 25 cents coin $\leq 3$
4. Number of 5 cents coins + number of 10 cents coins $\leq 2$
The Cashier algorithm is optimal for U.S. coin denomination (old Canadian coin system).

This algorithm does not always produce an optimal solution.

- Example:
  - US. post denomination
  - New Canadian coin system

- If the denominations include 1, then there is always a solution.

- Why the proof does not work for US. post denomination?
  - Left as an exercise