CS 341: ALGORITHMS

Lecture 8: greedy algorithms II
Readings: see website

Trevor Brown

https://student.cs.uwaterloo.ca/~cs341

trevor.brown@uwaterloo.ca
KNAPSACK PROBLEMS
Problem 4.4

Knapsack

Instance: Profits \( P = [p_1, \ldots, p_n] \); weights \( W = [w_1, \ldots, w_n] \); and a capacity, \( M \). These are all positive integers.

Feasible solution: An \( n \)-tuple \( X = [x_1, \ldots, x_n] \) where \( \sum_{i=1}^{n} w_i x_i \leq M \).

Gotta respect the weight limit…
Knapsack

Instance: Profits $P = [p_1, \ldots, p_n]$; weights $W = [w_1, \ldots, w_n]$; and a capacity, $M$. These are all positive integers.

Feasible solution: An $n$-tuple $X = [x_1, \ldots, x_n]$ where $\sum_{i=1}^{n} w_i x_i \leq M$.

In the 0-1 Knapsack problem (often denoted just as Knapsack), we require that $x_i \in \{0, 1\}$, $1 \leq i \leq n$.

In the Rational Knapsack problem, we require that $x_i \in \mathbb{Q}$ and $0 \leq x_i \leq 1$, $1 \leq i \leq n$.

Find: A feasible solution $X$ that maximizes $\sum_{i=1}^{n} p_i x_i$.

0-1 Knapsack: NP Hard. Probably requires exponential time to solve...

Rational knapsack: Can be solved in polynomial time by a greedy alg!

Lets discuss this now... other one later
POSSIBLE GREEDY STRATEGIES FOR KNAPSACK PROBLEMS

• **Strategy 1**: consider items in **decreasing** order of profit (i.e., we maximize the local evaluation criterion $p_i$)

• Let’s try an example input
  - Profits $P = [20, 50, 100]$
  - Weights $W = [10, 20, 10]$
  - Weight limit $M = 10$

• Algorithm selects last item for 100 profit
  - Looks optimal in this example
POSSIBLE GREEDY STRATEGIES FOR KNAPSACK PROBLEMS

• **Strategy 1**: consider items in **decreasing** order of **profit** (i.e., we maximize the local evaluation criterion $p_i$)

• How about a **second example input**
  • Profits $P = [20, 50, 100]$
  • Weights $W = [10, 20, 100]$
  • Weight limit $M = 10$

• Algorithm selects last item for **10 profit**
  • **Not optimal!**
POSSIBLE GREEDY STRATEGIES FOR KNAPSACK PROBLEMS

• **Strategy 2:** consider items in **increasing** order of **weight** (i.e., we minimize the local evaluation criterion $w_i$)

• **Counterexample**
  
  • Profits $P = [20, 50, 100]$
  • Weights $W = [10, 20, 100]$
  • Weight limit $M = 10$
  • Algorithm selects first item for 20 profit
    • It **could** select half of second item, for 25 profit!
POSSIBLE GREEDY STRATEGIES FOR KNAPSACK PROBLEMS

• **Strategy 3:** consider items in **decreasing** order of **profit divided by weight** (i.e., we maximize local evaluation criterion $p_i/w_i$)

• Let’s try our first example input
  
  • Profits $P = [20,50,100]$
  
  • Weights $W = [10,20,10]$
  
  • Weight limit $M = 10$
  
  • Profit divided by weight

  • $P/W = [2, 2.5, 10]$

  • Algorithm selects last item for 100 profit (optimal)
POSSIBLE GREEDY STRATEGIES FOR KNAPSACK PROBLEMS

• **Strategy 3:** consider items in **decreasing** order of profit divided by weight (i.e., we maximize local evaluation criterion $p_i/w_i$)

• Let’s try our second example input
  - Profits $P = [20, 50, \textbf{100}]$
  - Weights $W = [10, 20, 100]$
  - Weight limit $M = 10$
  - Profit divided by weight
    - $P/W = [2, 2.5, 1]$
  - Algorithm selects second item for 25 profit (optimal)

*It turns out strategy #3 is optimal...*
Preprocess(A[1..n], M) // A[i] = (p_i, w_i)
sort A by decreasing profit divided by weight
let p[1..n] be the profits in A
let w[1..n] be the weights in A
return GreedyRationalKnapsack(p, w, M)

GreedyRationalKnapsack(p[1..n], w[1..n], M)
X = [0, ..., 0]
weight = 0
for i = 1..n
    if weight + w[i] > M then
        X[i] = (M - weight) / w[i]
        break
    else
        X[i] = 1
        weight = weight + w[i]
return X

Either X=(1,1,...,1,0,...,0) or X=(1,1,...,1,x_i,0,...,0) where x_i ∈ (0,1)
Preprocess(A[1..n], M) // A[i] = (p_i, w_i)
sort A by decreasing profit divided by weight
let p[1..n] be the profits in A
let w[1..n] be the weights in A
return GreedyRationalKnapsack(p, w, M)

GreedyRationalKnapsack(p[1..n], w[1..n], M)
X = [0, ..., 0]
weight = 0

for i = 1..n
  if weight + w[i] > M then
    X[i] = (M - weight) / w[i]
    break
  else
    X[i] = 1
    weight = weight + w[i]

return X
INFORMAL FEASIBILITY ARGUMENT
(SHOULD BE GOOD ENOUGH TO SHOW FEASIBILITY ON ASSESSMENTS)

• Feasibility: all \( x_i \) are in \([0, 1]\) and total weight is \( \leq M \)
• Either everything fits in the knapsack, or:
• When we exit the loop, weight is exactly \( M \)
• Every time we write to \( x_i \) it’s either 0, 1 or \((M - \text{weight})/w_i\) where \( \text{weight} + w[i] > M \)
  • Rearranging the latter we get \((M - \text{weight})/w_i < 1\)
  • And \( \text{weight} \leq M \), so \((M - \text{weight})/w_i \geq 0\)
• So, we have \( x_i \in [0, 1] \)

```c
for i = 1..n
  if weight + w[i] > M then
    X[i] = (M - weight) / w[i]
    break
  else
    X[i] = 1
    weight = weight + w[i]
```
MINOR MODIFICATION TO FACILITATE FORMAL PROOF

```python
GreedyRationalKnapsack(p[1..n], w[1..n], M)
X = [0, ..., 0]
weight = 0

for i = 1..n
    if weight + w[i] > M then
        X[i] = (M - weight) / w[i]
        weight = M
        break
    else
        X[i] = 1
        weight = weight + w[i]

return X
```

Does NOT change behaviour of the algorithm at all!
FORMAL FEASIBILITY ARG

• Loop invariant: $\forall i : x_i \in [0,1]$
• and $\text{weight} = \sum_{i=1}^{n} w_i x_i \leq M$
• Base case. Initially $\text{weight} = 0$ and $\forall i : x_i = 0$.
  • So $0 = \text{weight} = \sum_{i=1}^{n} w_i \cdot 0 = \sum_{i=1}^{n} w_i x_i \leq M$
• Inductive step.
  • Suppose invariant holds at start of iteration $i$
  • Let $\text{weight}', x_i'$ denote values of $\text{weight}, x_i$ at end of iteration $i$
  • Prove invariant holds at end of iteration $i$
  • i.e., $\forall i : x_i' \in [0,1]$ and $\text{weight}' = \sum_{i=1}^{n} w_i x_i' \leq M$

```
for i = 1..n
  if \text{weight} + w[i] > M then
    X[i] = (M - \text{weight}) / w[i]
    \text{weight} = M
    break
  else
    X[i] = 1
    \text{weight} = \text{weight} + w[i]
```
FORMAL FEASIBILITY ARG

- **WTP**: \( \forall i : x'_i \in [0, 1] \)
  and \( \text{weight'} = \sum_{i=1}^{n} w_i x'_i \leq M \)

- **Case 1**: \( \text{weight} + w_i \leq M \)
  - \( x'_i = 1 \) **which is in** \([0, 1]\) **(by line 11)**
  - \( \text{weight'} = \text{weight} + w_i \) **(by line 12)**
    and **this is** \( \leq M \) by the case
  - \( \text{weight'} = \sum_{k=1}^{n} x_k w_k + w_i \) **(by invariant)**
  - \( \text{weight'} = \sum_{k=1}^{n} x_k w_k + x'_i w_i \) **(since** \( x'_i = 1 \)**)
  - And \( x'_k = x_k \) for all \( k \neq i \) and \( x_i = 0 \) so \( \sum_{k=1}^{n} x'_k w_k = x'_i w_i + \sum_{k=1}^{n} x_k w_k \)
  - Rearrange to get \( \sum_{k=1}^{n} x_k w_k = (\sum_{k=1}^{n} x'_k w_k - x'_i w_i) \)
  - So \( \text{weight'} = (\sum_{k=1}^{n} x'_k w_k - x'_i w_i) + x'_i w_i = \sum_{k=1}^{n} x'_k w_k \)
WTP: \( \forall i: x'_i \in [0, 1] \) and \( \text{weight}' = \sum_{i=1}^{n} w_i x'_i \leq M \)

Case 2: \( \text{weight} + w_i > M \)

- We have \( w_i > M - \text{weight} \) and \( M - \text{weight} \geq 0 \) (by case)
- So \( 0 \leq \frac{M - \text{weight}}{w_i} < 1 \) which means \( x'_i \in [0, 1) \)
- \( \text{weight}' = M = \text{weight} + (M - \text{weight}) \) (by line 8)
- \( \text{weight}' = \sum_{k=1}^{n} x_k w_k + (M - \text{weight}) \) (by invariant)
- But \( x'_k = x_k \) for all \( k \neq i \) and \( x_i = 0 \) so \( \sum_{k=1}^{n} x'_k w_k = x'_i w_i + \sum_{k=1}^{n} x_k w_k \)
- Rearrange to get \( \sum_{k=1}^{n} x_k w_k = (\sum_{k=1}^{n} x'_k w_k - x'_i w_i) \)
- So \( \text{weight}' = (\sum_{k=1}^{n} x'_k w_k - x'_i w_i) + (M - \text{weight}) \)
- And \( M - \text{weight} = x'_i w_i \) so \( \text{weight}' = \sum_{k=1}^{n} x'_k w_k \)
OPTIMALITY

For simplicity, assume that the profit / weight ratios are all distinct, so

\[
\frac{p_1}{w_1} > \frac{p_2}{w_2} > \cdots > \frac{p_n}{w_n}.
\]

Suppose the greedy solution is \( X = (x_1, \ldots, x_n) \) and the optimal solution is \( Y = (y_1, \ldots, y_n) \).

We will prove that \( X = Y \), i.e., \( x_j = y_j \) for \( j = 1, \ldots, n \). Therefore there is a unique optimal solution and it is equal to the greedy solution.

Suppose \( X \neq Y \).

Pick the smallest integer \( j \) such that \( x_j \neq y_j \).

To obtain a contradiction

\( X \) and \( Y \) are identical up to \( x_j \) and \( y_j \), respectively
What's the relationship between $x_j$ and $y_j$?
Can we have $y_j > x_j$?

No! Greedy would take more of item $j$ if it could.

$j = \text{first index where the solutions differ}$
fraction of item in knapsack

Greedy solution $X$

Optimal solution $Y$

$j = \text{first index where the solutions differ}$

Must have $y_j < x_j$

$(x_j - y_j)$
Greedy solution $X$

Optimal solution $Y$

$j =$ first index where the solutions differ

Can $Y$ be all zeros after $y_j$?

No! It would be worth less than $X$
Greedy solution X

Optimal solution Y

fraction of item in knapsack

x_1 x_2 \ldots x_{j-1} x_j

y_1 y_2 \ldots y_{j-1} y_j

j = first index where the solutions differ

Must exist k > j such that y_k > 0

Remove some of item k and replace it with some of item j?

How much of item k should we remove?

But, by our sort order, item j is worth more (per unit of weight) than item k!
Since item j is worth **more per unit weight**, replacing **even a tiny amount** of item k with item j will improve the solution.

So, we remove an infinitesimal $\delta > 0$ of weight of item k, and add $\delta$ weight of item j.
Greedy solution X

Optimal solution Y

j = first index where the solutions differ

To move $\delta$ weight from item $k$ to item $j$...

What fraction of item $k$ are we removing?

$\delta \frac{w_k}{y_k}$

What fraction of item $j$ are we adding?

$\delta \frac{w_j}{y_j}$

Modified optimal solution $Y'$

$y'_j = y_j + \delta \frac{w_j}{y_j}$

$y'_k = y_k - \delta \frac{w_k}{y_k}$
The idea is to show that
\[ Y' \text{ is feasible, and} \]
\[ \text{profit}(Y') > \text{profit}(Y). \]
This contradicts the optimality of \( Y \) and proves that \( X = Y \).

To show \( Y' \) is feasible, we show \( y'_k \geq 0, y'_j \leq 1 \) and \( \text{weight}(Y') \leq M \).
**FEASIBILITY OF $Y'$**

- To show $Y'$ is feasible, we show $y'_k \geq 0, y'_j \leq 1$ and $weight(Y') \leq M$

- Let's show $y'_k \geq 0$
  
  - By definition, $y'_k = y_k - \frac{\delta}{w_k}$
  
  - So, $y'_k \geq 0$ iff $y_k - \frac{\delta}{w_k} \geq 0$ iff $\delta \leq y_kw_k$
  
  - And $y_k$ and $w_k$ are both **positive**
  
  - So, this constrains $\delta$ to be smaller than a **positive number**
  
  - Therefore, it is possible to choose positive $\delta$ s.t. $y'_k \geq 0$
FEASIBILITY OF $Y'$

• To show $Y'$ is feasible, we show $y'_k \geq 0$, $y'_j \leq 1$ and $\text{weight}(Y') \leq M$

• Now let's show $y'_j \leq 1$

  • By definition, $y'_j = y_j + \frac{\delta}{w_j}$

  • So, $y'_j \leq 1$ iff $y_j + \frac{\delta}{w_j} \leq 1$ iff $\delta \leq (1 - y_j)w_j$

  • Recall $y_j < x_j$, so $y_j < 1$, which means $(1 - y_j) > 0$

  • So, this constrains $\delta$ to be smaller than some positive number
FEASIBILITY OF $Y'$

- Finally, we show $\text{weight}(Y') \leq M$

- Recall changes to get $Y'$ from $Y$
  - We move $\delta$ weight from item $k$ to item $j$
  - This does not change the total weight!
  - So $\text{weight}(Y') = \text{weight}(Y) \leq M$
  - Therefore, $Y'$ is feasible!
SUPERIORITY OF Y’

• Finally we compute \( \text{profit}(Y') \)

\[
\text{profit}(Y') = \text{profit}(Y) + \frac{\delta}{w_j}p_j - \frac{\delta}{w_k}p_k
\]

\[
= \text{profit}(Y) + \delta \left( \frac{p_j}{w_j} - \frac{p_k}{w_k} \right)
\]

• Since j is before k, and we consider items with more profit per unit weight first, we have \( \frac{p_j}{w_j} > \frac{p_k}{w_k} \).

• So, if \( \delta > 0 \) then \( \delta \left( \frac{p_j}{w_j} - \frac{p_k}{w_k} \right) > 0 \)

• Since we can choose \( \delta > 0 \), we have \( \text{profit}(Y') > \text{profit}(Y) \).
PROBLEM: COIN CHANGING
Problem 4.5

Coin Changing

Instance: A list of coin denominations, $d_1, d_2, \ldots, d_n$, and a positive integer $T$, which is called the target sum.

Find: An $n$-tuple of non-negative integers, say $A = [a_1, \ldots, a_n]$, such that $T = \sum_{i=1}^{n} a_i d_i$ and such that $N = \sum_{i=1}^{n} a_i$ is minimized.

In the **Coin Changing** problem, $a_i$ denotes the number of coins of denomination $d_i$ that are used, for $i = 1, \ldots, n$.

The total value of all the chosen coins must be exactly equal to $T$. We want to **minimize** the number of coins used, which is denoted by $N$. 
EXAMPLE: CANADIAN COINS (R.I.P. PENNY)
EXAMPLE: CANADIAN COINS

• Input: coin denominations = 200, 100, 25, 10, 5, 1 (R.I.P.)
  target sum $T = 155$

• Output: **minimum number** of coins to pay $T$
  (and list of coins)

• Solution: $1 \times 100 + 2 \times 25 + 1 \times 5$ ; 4 coins

• Suggestion for an algorithm?
  • Sort coin denominations from largest to smallest value
  • Greedily use the largest possible coin at all times
GreedyCoinChanging(D[1..n], T)

1. sort D in decreasing order
2. used = [0, ..., 0]

for i = 1..n
1. used[i] = floor(T / D[i])
2. T = T - (used[i] * D[i])

if T > 0 then return FAIL

return used

Maybe it's impossible to obtain T (explicitly represent impossibility of a feasible solution)

Feasibility proof is trivial. Either we achieve T or return FAIL.

Runtime complexity?

used[i] = # of coins used of type D[i]

Try all coin types

Use as many of coins of type D[i] as we can

Update the target sum to account for these coins
OPTIMALITY

• Is this algorithm optimal?
• Trying to build a correctness argument:
  • Fix part of the input:
    • Canadian coin system (including pennies)
  • Try to prove optimality for all target sums T
• Reasoning about one class of inputs at a time can make an algorithm easier to understand
We will prove that the greedy algorithm always finds an optimal solution for coin denominations $D = [100, 25, 10, 5, 1]$.

We will make use of the following properties of any optimal solution:

(1) the number of pennies is at most 4 (replace five pennies by a nickel)

(2) the number of nickels is at most 1 (replace two nickels by a dime)

(3) the number of quarters is at most 3 (replace four quarters by a loonie), and

(4) the number of nickels + the number of dimes is at most 2 (replace three dimes by a quarter and a nickel; replace two dimes and a nickel by a quarter; the number of nickels is at most one).

The proof is by induction on $T$. As (trivial) base cases, we can take $T = 1, 2, 3, 4$. 
Inductive step ($T > 4$): assume greedy makes optimal change for target values less than $T$. Show it makes optimal change for $T$.

Suppose $5 \leq T < 10$. First, assume there is no nickel in the optimal solution. Then the optimal solution contains only of pennies, so $T \leq 4$ (property (1)); contradiction. Therefore the optimal solution contains at least one nickel. Clearly the greedy solution contains at least one nickel. By induction, the greedy solution for $T - 5$ is optimal. Therefore the greedy solution for $T$ is also optimal.

(1) the number of pennies is at most 4 (replace five pennies by a nickel)
(2) the number of nickels is at most 1 (replace two nickels by a dime)
(3) the number of quarters is at most 3 (replace four quarters by a loonie), and
(4) the number of nickels + the number of dimes is at most 2 (replace three dimes by a quarter and a nickel; replace two dimes and a nickel by a quarter; the number of nickels is at most one).
Suppose $10 \leq T < 25$. First, assume there is no dime in the optimal solution. Then the optimal solution contains only nickels and pennies, so $T \leq 5 + 4 = 9$ (property (2)); contradiction. Therefore the optimal solution contains at least one dime. Clearly the greedy solution contains at least one dime. By induction, the greedy solution for $T - 10$ is optimal. Therefore the greedy solution for $T$ is also optimal.

(1) the number of pennies is at most 4 (replace five pennies by a nickel)
(2) the number of nickels is at most 1 (replace two nickels by a dime)
(3) the number of quarters is at most 3 (replace four quarters by a loonie), and
(4) the number of nickels + the number of dimes is at most 2 (replace three dimes by a quarter and a nickel; replace two dimes and a nickel by a quarter; the number of nickels is at most one).
Recall: proof for $10 \leq T < 25$

Suppose $10 \leq T < 25$. First, assume there is no dime in the optimal solution. Then the optimal solution contains only nickels and pennies, so $T \leq 5 + 4 = 9$ (property (2)); contradiction. Therefore the optimal solution contains at least one dime. Clearly the greedy solution contains at least one dime. By induction, the greedy solution for $T - 10$ is optimal. Therefore the greedy solution for $T$ is also optimal.

(1) the number of pennies is at most 4 (replace five pennies by a nickel)
(2) the number of nickels is at most 1 (replace two nickels by a dime)
(3) the number of quarters is at most 3 (replace four quarters by a loonie), and
(4) the number of nickels + the number of dimes is at most 2 (replace three dimes by a quarter and a nickel; replace two dimes and a nickel by a quarter; the number of nickels is at most one).

Recall: properties of any optimal solution
• Exercise: suppose $25 \leq T < 100$
  • Find one coin that must be in optimal & greedy to reduce this case to making change for less than $T$
  • Assume no quarters in optimal solution
    • Then by properties 1&4, the optimal solution uses at most:
      (4 pennies) and (2 nickels or dimes)
    • Max value is therefore 24 cents, so cannot make $T$ change!
  • So optimal contains a quarter. (And so does greedy.)
  • By inductive hypothesis, greedy is optimal for $T - 25$.
  • So, greedy is optimal for $T$. 
• Exercise: suppose $100 \leq T < 200$
  • Find one coin that must be in optimal & greedy to reduce this case to making change for less than $T$
  • Assume no loonies in optimal solution
    • Then by properties 1, 3, 4, the optimal solution uses at most: (4 pennies) and (2 nickels or dimes) and (3 quarters)
    • Max value is therefore 99 cents, so cannot make $T$ change!
  • So optimal contains a loonie. (And so does greedy.)
  • By inductive hypothesis, greedy is optimal for $T - 100$.
  • So, greedy is optimal for $T$.
• Exercise for outside lecture: $200 \leq T$
WHAT ABOUT OTHER COIN SYSTEMS?

• Optimal for old Canadian coin system
• How about new Canadian coin system?
  • Denominations: 200, 100, 25, 10, 5
  • Some values can’t be created at all!
• How about the old British coin system
  • Denominations: 30, 24, 12, 6, 3, 1
  • Counter-example: T=48. Greedy=30,12,6 ; Opt=24,24
• What makes a coin system optimal / non-optimal?
MORE CHALLENGING HOME EXERCISE:

• Show greedy is optimal for any coin system satisfying:
  • \( d_j \mid d_{j-1} \) for all \( j, 2 \leq j \leq n \)
  • Hints (tiny font, so no spoilers):

• Is greedy non-optimal for every coin system that does not satisfy this property?
  • No, it’s optimal for old Canadian coins even though 10 does not divide 25
  • So, the above condition is sufficient but not necessary