Approximation Algorithms

Knapsack

Outline

- Approximation algorithms
  - Exact vs approximate solution
- The Knapsack problem
  - Greedy 2-approximation algorithm

Outline

- Approximation algorithms
  - Exact vs approximate solution
- The Knapsack problem
  - Greedy 2-approximation algorithm

Learning objectives:
You are able to
- describe formally and informally the definition of an approximation algorithm.
- design and analyse a 2-approximation algorithm to the Knapsack problem.

Approximation Algorithms

Exact solutions:

- Find the *minimum* spanning tree
- The *largest* number of sets that add up to T
- The *maximum* cardinality independent set
- …

Approximation Algorithms

Minimize x	**Objective function $\boldsymbol{f(x)}$**
Subject to	**Constraints** $\boldsymbol{g_1(x)}$ $\boldsymbol{g_2(x)}$ **…** $\boldsymbol{g_m}$

Approximation Algorithms

Minimize x	**Objective function $f(x)$**
Subject to	**Constraints** $g_1(x)$, $g_2(x)$, …, g_m

Find the optimum solution OPT with the cost cost(OPT)

Approximation Algorithms

Minimize x **Objective function $f(x)$**

Subject to **Constraints** $g_1(x)$, $g_2(x)$, …, g_m

Find the optimum solution OPT with the cost cost(OPT)

Sometimes it's much easier to find a solution S that is *almost* optimal.

Of course, S should still satisfy the constraints.

Approximation Algorithms

Minimize x

Objective function $f(x)$

Find the optimum solution OPT with the cost cost(OPT)

Subject to

Constraints $g_1(x)$ $g_2(x)$ … g_m

E.g., T is a spanning tree of the input graph G.

Sometimes it's much easier to find a solution S that is *almost* optimal.

Of course, S should still satisfy the constraints.

Approximation Algorithms

An algorithm A is called an α-approximation algorithm for a minimization problem P, if for every input I, the cost $\text{cost}(A(I))$ satisfies

$$\text{cost}(A(I)) \leq \text{cost}(\text{OPT}(I)) \cdot \alpha$$

Approximation Algorithms

An algorithm A is called an α-approximation algorithm for a minimization problem P, if for every input I, the cost $\text{cost}(A(I))$ satisfies

$$\text{cost}(A(I)) \leq \text{cost}(\text{OPT}(I)) \cdot \alpha$$

Demand $\text{cost}(A(I)) \geq \text{cost}(\text{OPT}(I)) \,/\alpha$
in case of maximization

Approximation Algorithms

An algorithm A is called an α-approximation algorithm for a minimization problem P, if for every input I, the cost cost($A(I)$) satisfies

$$\mathrm{cost}(A(I)) \leq \mathrm{cost}(\mathrm{OPT}(I)) \cdot \alpha$$

Multiplicative approximation

Demand cost($A(I)$) $\geq$ cost(OPT(I)) $/\alpha$
in case of maximization

Approximation Algorithms

An algorithm A is called an α-approximation algorithm for a minimization problem P, if for every input I, the cost $\text{cost}(A(I))$ satisfies

$$\text{cost}(A(I)) \leq \text{cost}(\text{OPT}(I)) \cdot \alpha$$

Multiplicative approximation

Demand $\text{cost}(A(I)) \geq \text{cost}(\text{OPT}(I))\ /\alpha$ in case of maximization

Approximation Algorithms

An algorithm A is called an α-approximation algorithm for a minimization problem P, if for every input I, the cost cost$(A(I))$ satisfies

$$\text{cost}(A(I)) \leq \text{cost}(\text{OPT}(I)) \cdot \alpha$$

Demand cost$(A(I)) \geq$ cost$(\text{OPT}(I))\ /\alpha$
in case of maximization

Nicer to always say "α" approximation and not "$1/\alpha$" approximation.

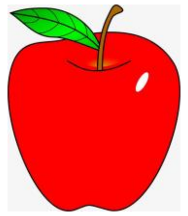
Outline

- Approximation algorithms
    - Exact vs approximate solution
- The Knapsack problem
    - Greedy 2-approximation algorithm

The Knapsack Problem

Sauerkraut
Value v_1: 1
Weight w_1: 7

Chocolate
Value v_2: 4
Weight w_2: 3

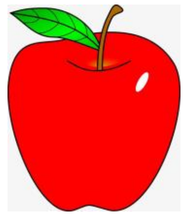


Apple
Value: v_3: 5
Weight: w_3: 4

The Knapsack Problem

Sauerkraut
Value v_1: 1
Weight w_1: 7

Chocolate
Value v_2: 4
Weight w_2: 3



Apple
Value: v_3: 5
Weight: w_3: 4

Capacity C

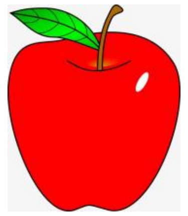
Maximize sum of values

Sum of weights at most C

The Knapsack Problem

Sauerkraut
Value v_1: 1
Weight w_1: 7

Chocolate
Value v_2: 4
Weight w_2: 3



Apple
Value: v_3: 5
Weight: w_3: 4

Set of items I
Unbounded Knapsack:
You can pick each item as many times as you want.

Capacity C

Maximize S $\quad \boxed{\Sigma_{S \subseteq I} V(S)}$

Subject to $\quad \boxed{W(S) \leq C}$

$V(S)$: sum of values
$W(S)$: sum of weights

The Greedy Algorithm

Capacity C

Maximize S $\Sigma_{S \subseteq I} V(S)$

Subject to $W(S) \leq C$

Theorem:
There is 2-approximation algorithm to the Unbounded Knapsack problem.

The Greedy Algorithm

Marginal gain:
Ratio of weight and value v_i / w_i

Sauerkraut
Value v_1: 1
Weight w_1: 7

Chocolate
Value v_2: 4
Weight w_2: 3

The Greedy Algorithm

Marginal gain:
Ratio of weight and value v_i/w_i

Sauerkraut
Value v_1: 1
Weight w_1: 7

Marginal gain: $1/7$

Chocolate
Value v_2: 4
Weight w_2: 3

The Greedy Algorithm

Marginal gain:
Ratio of weight and value v_i / w_i

Sauerkraut
Value v_1: 1
Weight w_1: 7

Marginal gain: $1/7$

Chocolate
Value v_2: 4
Weight w_2: 3

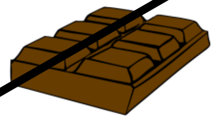
Marginal gain: $4/3$

The Greedy Algorithm

Sauerkraut
Value v_1: 1
Weight w_1: 7

Marginal gain:
Ratio of weight and value v_i/w_i

Marginal gain: $1/7$



Chocolate
Value v_2: 4
Weight w_2: 3

Chocolate is much better than sauerkraut.

Marginal gain: $4/3$

The Greedy Algorithm

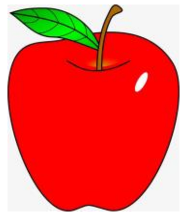
Marginal gain:
Ratio of weight and value v_i/w_i

Greedy algorithm

Sort items descending according to gain
Set $S = \emptyset$
While(∃ item $\boldsymbol{i}$ s.t. $\boldsymbol{W(S \cup i) \leq C}$)
  Let j be the item with the largest gain such that $W(S \cup j) \leq C$
  Set $S := S \cup j$

Sauerkraut
Value v_1: 1
Weight w_1: 7
Gain g_1: 1/7

Chocolate
Value v_2: 4
Weight w_2: 3
Gain g_2: 4/3



Apple
Value: v_3: 5
Weight: w_3: 4
Gain: g_3: 5/4

The Greedy Algorithm

Marginal gain:
Ratio of weight and value v_i/w_i

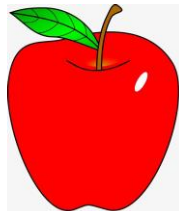
Greedy algorithm

Sort items descending according to gain
Set $S = \emptyset$
While(∃ item $\boldsymbol{i}$ s.t. $\boldsymbol{W(S \cup i) \leq C}$)
  Let j be the item with the largest gain such that $W(S \cup j) \leq C$
  Set $S := S \cup j$

Pick the best item that fits.

Sauerkraut
Value v_1: 1
Weight w_1: 7
Gain g_1: 1/7

Chocolate
Value v_2: 4
Weight w_2: 3
Gain g_2: 4/3



Apple
Value: v_3: 5
Weight: w_3: 4
Gain: g_3: 5/4

The Greedy Algorithm - Example

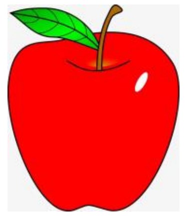
Let $C = 8$

Sauerkraut

Value	v_1:	1
Weight	w_1:	7
Gain	g_1:	1/7

Chocolate

Value	v_2:	4
Weight	w_2:	3
Gain	g_2:	4/3



Apple

Value:	v_3:	5
Weight:	w_3:	4
Gain:	g_3:	5/4

The Greedy Algorithm - Example

Let $C = 8$

Greedy: 8


Sauerkraut

Value v_1: 1

Weight w_1: 7

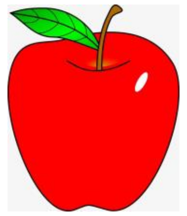
Gain g_1: 1/7

Chocolate

Value v_2: 4

Weight w_2: 3

Gain g_2: 4/3



Apple

Value: v_3: 5

Weight: w_3: 4

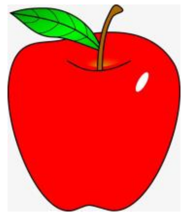
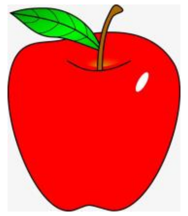
Gain: g_3: 5/4

The Greedy Algorithm - Example

Let $C = 8$

Greedy: 8


OPT: 10

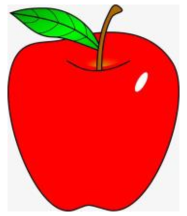



Sauerkraut

Value	v_1:	1
Weight	w_1:	7
Gain	g_1:	1/7

Chocolate

Value	v_2:	4
Weight	w_2:	3
Gain	g_2:	4/3



Apple

Value:	v_3:	5
Weight:	w_3:	4
Gain:	g_3:	5/4

The Greedy Algorithm

Theorem:
There is 2-approximation algorithm to the Unbounded Knapsack problem.

Suppose w.l.o.g. that, for all $i, w_i \leq C$, that is, each item fits into the knapsack.

The Greedy Algorithm

Theorem:
There is 2-approximation algorithm to the Unbounded Knapsack problem.

Suppose w.l.o.g. that, for all $i, w_i \leq C$, that is, each item fits into the knapsack.

Plan:

1. Let $x \in I$ be the item with the best marginal gain. If it was the case that C is a multiple of w_x, the best thing to do is to only choose items x.

The Greedy Algorithm

Theorem:
There is 2-approximation algorithm to the Unbounded Knapsack problem.

Suppose w.l.o.g. that, for all $i, w_i \leq C$, that is, each item fits into the knapsack.

Plan:

1. Let $x \in I$ be the item with the best marginal gain. If it was the case that C is a multiple of w_x, the best thing to do is to only choose items x.

2. Adding one more x to greedy must be better than OPT.

The Greedy Algorithm

Theorem:
There is 2-approximation algorithm to the Unbounded Knapsack problem.

Suppose w.l.o.g. that, for all $i, w_i \leq C$, that is, each item fits into the knapsack.

Plan:

1. Let $x \in I$ be the item with the best marginal gain. If it was the case that C is a multiple of w_x, the best thing to do is to only choose items x.

2. Adding one more x to greedy must be better than OPT.

3. One more x can at most double the value of the greedy solution.

The Greedy Algorithm

Input:
Set I of items, capacity C and let x be the item with the best marginal gain.

Denote greedy$(I) = S$ and OPT$(I) = S^*$

The Greedy Algorithm

Input:
Set I of items, capacity C and let x be the item with the best marginal gain.

Denote greedy$(I) = S$ and OPT$(I) = S^*$

Observation:

$$V(S^*) = \sum_{i \in S^*} v_i = \sum_{i \in S^*} \frac{w_i \cdot v_i}{w_i}$$

The Greedy Algorithm

Input:
Set I of items, capacity C and let x be the item with the best marginal gain.

Denote greedy$(I) = S$ and OPT$(I) = S^*$

Observation:

$$V(S^*) = \sum_{i \in S^*} v_i = \sum_{i \in S^*} \frac{w_i \cdot v_i}{w_i}$$

$$\leq \sum_{i \in S^*} \frac{w_i \cdot v_x}{w_x} \leq C \cdot \frac{v_x}{w_x}$$

The Greedy Algorithm

Observation:
It holds that $V(S) \geq v_x$.

Input:
Set I of items, capacity C and let x be the item with the best marginal gain.

Denote greedy$(I) = S$ and OPT$(I) = S^*$

The Greedy Algorithm

Input:
Set I of items, capacity C and let x be the item with the best marginal gain.

Denote greedy$(I) = S$ and OPT$(I) = S^*$

Observation:
It holds that $V(S) \geq v_x$.

Observation:
Suppose that there are k items x in S.
Then $(k+1) \cdot v_x \leq 2 \cdot V(S)$.

The Greedy Algorithm

Input:
Set I of items, capacity C and let x be the item with the best marginal gain.

Denote greedy$(I) = S$ and OPT$(I) = S^*$

Observation:
It holds that $V(S) \geq v_x$.

Observation:
Suppose that there are k items x in S. Then $(k+1) \cdot v_x \leq 2 \cdot V(S)$.

Proof:
By assumption, we have that
$V(S) \geq k \cdot v_x = (k+1) \cdot v_x - v_x$.
From the previous observation, we derive
$V(S) \geq (k+1) \cdot v_x - V(S)$ and
re-writing gives

$$2 \cdot V(S) \geq (k+1) \cdot v_x$$

The Greedy Algorithm

Claim:
Greedy algorithm gives a 2-approximation to the bounded Knapsack problem.

Observation:
Suppose that there are k items x in S. Then $(k+1) \cdot v_x \leq 2 \cdot V(S)$ and C $<$ $w_x \cdot (k+1)$

Observation:

$$V(S^*) \leq C \cdot \frac{v_x}{w_x}$$

The Greedy Algorithm

Claim:
Greedy algorithm gives a 2-approximation to the bounded Knapsack problem.

Proof:
Recall that S is the greedy and S^* the optimal solution, respectively. We have

$$V(S^*) \leq C \cdot \frac{v_x}{w_x} < \frac{w_x \cdot (k+1) \cdot v_x}{w_x}$$
$$= (k+1) \cdot v_x \leq 2 \cdot V(S)$$

Observation:
Suppose that there are k items x in S. Then $(k+1) \cdot v_x \leq 2 \cdot V(S)$ and C $< w_x \cdot (k+1)$

Observation:

$$V(S^*) \leq C \cdot \frac{v_x}{w_x}$$

The Greedy Algorithm

Theorem:
There is 2-approximation algorithm to the Unbounded Knapsack problem.

The Greedy Algorithm

Theorem:
There is 2-approximation algorithm to the Unbounded Knapsack problem.

Why does this not work in the bounded case?

Wrap-Up

Approximation:
Optimal solutions vs almost optimal solutions.

Knapsack:
The greedy algorithms is a 2-approximation