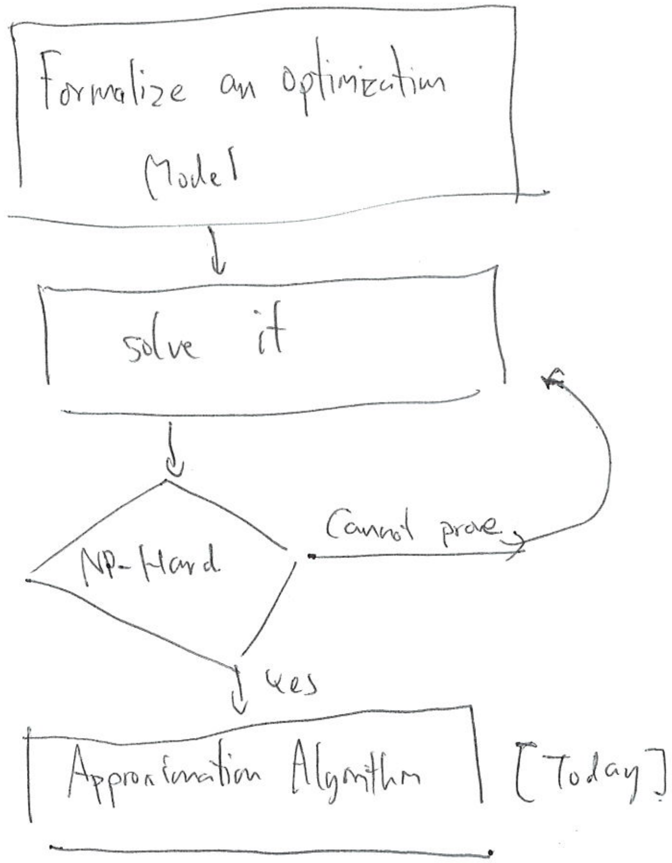
Approximation and Online Algorithms

with Applications

# 3.

Recap




---

## Knapsack Problem (Simplified Problem)

- We are in All-you-can-eat Strawberry farm.
- We can eat a limited amount of the strawberries, but we want to have it as much as possible.

Optimization Model

| Maximum Amount of strawberry we can take $W$.

Input: #strawberries $n$,
$w_1, \dots, w_n \in \mathbb{R}_{\geq 0}$ where $w_i$ is weight of strawberry $i$.
$w_1 \leq w_2 \leq \dots \leq w_n$. $w_i \leq W$

Output: $S \subseteq \{1, \dots, n\}$ set of strawberries we will take.

Constraint: $\sum_{i \in S} w_i \leq W$ summation of all weights in $S$ is ~~smaller~~ no larger than $W$.

Objective function: Maximize $\sum_{i \in S} w_i$

**Example** $n=5$, $W=10$.

$w_1=2$, $w_2=5$, $w_3=5$, $w_4=6$, $w_5=9$.

**optimal solution** : $S=\{2,3\} \rightarrow S^*$

**Optimal Value** : $w_2+w_3=10 \longrightarrow OPT$.

---

This problem is NP-Hard problem!

**Algorithm**

```
S ← φ
for j=1 to n:
    if Σ_{i∈S} w_i + w_j ≤ W:
        S ← S ∪ {j}
    else:
        if w_j ≥ Σ_{i∈S} w_i:
            S ← {j}
        break
```

---

**Example**

**Step1** $S=\phi$, $\sum_{i\in S} w_i = 0$, $j=1$

$$\sum_{i\in S} w_i + w_j = 0+2 = 2 \leq W \quad (10)$$

$$S \leftarrow \emptyset \cup \{1\} = \{1\}$$

Step2 $\quad S=\{1\}, \quad \sum_{i\in S} w_i = 2, \quad j=2$

$$\sum_{i\in S} w_i + w_j = 2+5 = 7 \leq W \qquad (10)$$

$$S = \{1\} \cup \{2\} = \{1,2\}.$$

Step3 $\quad S=\{1,2\}, \quad \sum_{i\in S} w_i = 7, \quad j=3.$

$$\sum_{i\in S} w_i + w_j = 7+5 = 12 > W.$$

Because $\sum_{i\in S} w_i > w_j$, we do not update $S$ at Line 7.

---

Optimal Solution: $S^* = \{2,3\}$ $\qquad$ Our solution: $S=\{1,2\}$

Optimal Value: $OPT := w_2 + w_3 = 10.$ $\qquad$ Our objective value:

$$SOL := w_1 + w_2 = 2+5 = 7.$$

Theorem: For any input, $SOL \geq \frac{1}{2} OPT$ $\longrightarrow$ Approximation ratio.

Thus, our algorithm is an $\frac{1}{2}$-approximation algorithm.

Proof:

Case1: $\sum_{i\in S} w_i \leq W$: Then, $SOL = W = OPT$

$$SOL \geq \frac{1}{2} \cdot OPT.$$

Case2: $\sum_{i\in S} w_i + w_j > W \longrightarrow \sum_{i\in S} w_i$ or $w_j \geq \frac{W}{2}$

$$\max\left\{\sum_{i\in S} w_i, w_j\right\} \geq \frac{W}{2}$$

By Lines 6-7, we select $S \leftarrow \{j\}$ when $w_j \geq \sum_{i \in S} w_i$.

When the algorithm terminates, $\sum_{i \in S} w_i = \max\{\sum_{i \in S} w_i, w_j\} \geq \frac{W}{2} \cancel{\geq \frac{OPT}{2}}$.

$$SOL = \max\{\sum_{i \in S} w_i, w_j\} \geq \frac{W}{2} \geq \frac{OPT}{2}$$

$W \geq OPT$

$$\therefore SOL \geq \frac{OPT}{2} \quad \boxed{\text{Q.E.D.}}$$

---

## Knapsack Problem (General Version)

- Each strawberry tastes differently, and our happiness from each strawberry does not depends on weight.
- We want to maximize our happiness.

### Optimization Model (OPT)

**Input:** # strawberries $n$,

$w_1, \ldots, w_n \in \mathbb{R}_{\geq 0}$ : $w_i$ weight of strawberry $i$.

$h_1, \ldots, h_n \in \mathbb{R}_{\geq 0}$ : $h_i$ happiness from strawberry $i$.

Assumption : $\frac{w_1}{h_1} \leq \frac{w_2}{h_2} \leq \ldots \leq \frac{w_n}{h_n}$.

$\rightarrow \frac{h_1}{w_1} \geq \frac{h_2}{w_2} \geq \ldots \geq \frac{h_n}{w_n}$.

$\rightarrow$ happiness we have for a unit of weight.

$W$: maximum weight.

**Output:** $S \subseteq \{1, \ldots, n\}$ : selected strawberries.

**Constraint:** $\cancel{|S|}$ $\sum_{i \in S} w_i \leq W$. **Objective Function:** Maximize $\sum_{i \in S} h_i$

More general problem? (R)

Input: Same

Output: $x_1, \ldots, x_n \in [0,1]$ : portion of strawberry $i$ we will take.

Constraint: $w_1 x_1 + w_2 x_2 + \ldots + w_n x_n = \sum_{i=1}^{n} w_i x_i \leq W$

Objective Function: Maximize $h_1 x_1 + h_2 x_2 + \ldots + h_n x_n = \sum_{i=1}^{n} h_i x_i$.

---

Optimal algorithm for R

$OPT_I \leq OPT_R$

```
x_j ← 0  for all j,  w ← 0
for j = 1 to n:
    if w + w_j ≤ W:
        x_j ← 1
        w ← w + w_j
    else
        x_j ← (W - w) / w_j
        break.
```

$OPT_I \leq OPT_R \leq h_1 + \ldots + h_j$

$\longrightarrow$ ~~$OPT_I \leq w_1 + \ldots + w_j$~~

---

Approximation Algorithm for I

```
S ← ∅ , w ← 0
for j = 1 to n:
    if w + w_j ≤ W:
        S ← S ∪ {j}
        w ← w + w_j
    else
        if W < w_j:
            S ← {j}
        break
```

$SOL = h_1 + \ldots + h_{j-1}$

$SOL = h_j$

$SOL = \max\{h_1 + \ldots + h_{j-1}, h_j\}$

$OPT_I \leq \underbrace{h_1 + \ldots + h_{j-1}} + h_j$

We know that $h_1 + \dots + h_{j-1}$ or $h_j \geq OPT_I/2$

$$SOL = \max\{h_1 + \dots + h_{j-1}, h_j\} \geq OPT_I/2$$

$$SOL \geq OPT_I/2$$

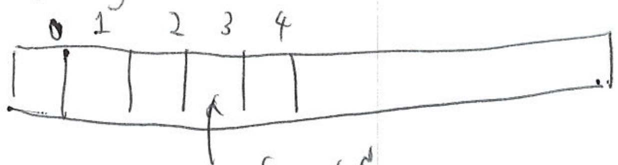
<u>Theorem</u>: The algorithm is an $1/2$-approximation algorithm for the knapsack problem.

---

<u>Bloom Filters</u> [Zhong, Lu, Shen, Seiferas, PODC '08] → Principles of Distributed Computing.

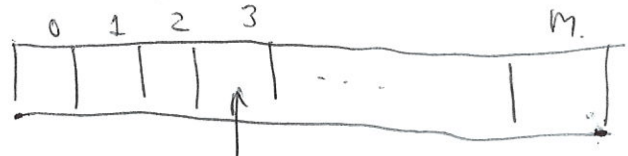
Data structure for set $\{s_1, \dots, s_n\} = S$

- Classical Data Structures.
  - Take a lot of times for checking membership $s \in S$
- Bit Array




0 if $s \notin S$
1 if $s \in S$

  - Fast search
  - Large memory when $s_i \in [0, 100{,}000{,}000]$.
- Hash Array. $H : S \to [1, m]$




1 if there is $s_i$ such that $H(s_i) = 3$.
0 otherwise.

Membership check $s \in S$? → Yes, if array at $H(s) = 1$.
No, otherwise.

$s \in S \rightarrow$ array of position $H(s) = 1 \rightarrow$ Always yes.

$s \notin S \rightarrow$ array of position $H(s)$ might be $1$ $\rightarrow$ May be yes

if there is $s' \in S$ such that (false positive)

$H(s') = H(s)$

Assume that for each $s'$, $Pr\{H(s') = \cancel{H(s)}\ 3\} = \frac{1}{m}$.

$Pr\{H(s') = H(s)\} = \frac{1}{m}$.

Then, Probability that each $s'$ has $H(s') \neq H(s)$ is $1 - \frac{1}{m}$.

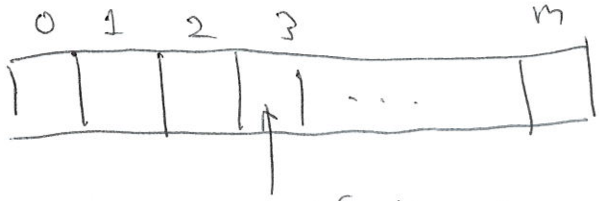
probability that ~~no~~ all $s'$ has $H(s') \neq H(s)$ / no $s'$ has $H(s') = H(s)$ is $\left(1 - \frac{1}{m}\right)^n$,

propability that there is $s'$ has $H(s') = H(s)$ is $1 - \left(1 - \frac{1}{m}\right)^n$,

---

- Bloom Filter $H_1 : S \rightarrow [1, m]$, $H_2 : S \rightarrow [1, m]$, $\ldots$ $H_k : S \rightarrow [1, m]$

(each hash give different value for each $s_i$)




$1$ if there is $s_i$ and $j$ such that $H_j(s_i) = 3$,

$0$ otherwise.

$s \in S$? $\rightarrow$ Yes if array at $H_j(s) = 1$ for all $j$.

$\rightarrow$ No, otherwise.

$s \in S \rightarrow$ array of positions $H_j(s) = 1$ for all $j$ $\rightarrow$ Always yes.

$s \notin S \rightarrow$ For all $j$, there is $s'$ and $j'$ such that $\rightarrow$ May be yes.

$$H_{j'}(s') = H_j(s)$$

- Larger $k$
  - Can check more cells $\rightarrow$ small false positive chance.
  - Larger # 1. $\rightarrow$ larger ~~chance~~ chance for false positive

Optimal $k$?

Probability that each $j'$ and $s'$ has $H_{j'}(s') \neq H_j(s)$ is

$$1 - \frac{1}{m}$$

" ———— " all $j'$ and $s'$ has $H_{j'}(s') \neq H_j(s)$ is
no $j'$ and $s'$ has $H_{j'}(s') = H_j(s)$ $\left(1 - \frac{1}{m}\right)^{kn}$.

" ———— " some $j'$ and $s'$ has $H_{j'}(s') = H_j(s)$ is

$$1 - \left(1 - \frac{1}{m}\right)^{kn}.$$

For all $j$: " ———— "

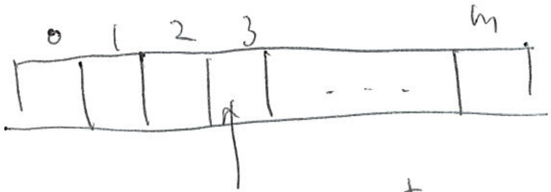
$$\left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k.$$

$\left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k$ is minimized when $k = \frac{m}{n} \cdot \ln 2$.

# Optimization Model

- Each s has different probability to be in S.
- Can we use that information to further reduce the false positive?
- Can we have different k for each s?

$S \subseteq \{s_1, \dots, s_n\}$ $\qquad$ $s_i$ will assign 1 to $k_i$ cells of the array.



Pr that $s_1$ will not assign 1 is $\left(1-\frac{1}{m}\right)^{k_1}$

— " — $s_2$ will not assign 1 is $\left(1-\frac{1}{m}\right)^{k_2}$

⋮

— " — $s_n$ — " — is $\left(1-\frac{1}{m}\right)^{k_n}$.

- Pr. that no element assign 1 is $\left(1-\frac{1}{m}\right)^{\sum_{i\in S} k_i}$ [Pr that the cell is 0]

Fact: False positive is minimized when

$$\boxed{\left(1-\frac{1}{m}\right)^{\sum_{i\in S} k_i} = \frac{1}{2}} \rightarrow \ln\left(\left(1-\frac{1}{m}\right)^{\sum_{i\in S} k_i}\right) = \ln\frac{1}{2}$$

$1-x \cong e^{-x}$ when x is small

$$\sum_{i\in S} k_i \ln\left(1-\frac{1}{m}\right) = \ln\left(\frac{1}{2}\right)$$

$$\sum_{i\in S} k_i \ln\left(e^{-\frac{1}{m}}\right) = -\ln 2$$

$$\sum_{i\in S} k_i \left(\cancel{-}\frac{1}{m}\right) = \cancel{-}\ln 2$$

$$\boxed{\sum_{i\in S} k_i = m \cdot \ln 2}$$ Constraint?

$$\cancel{\sum_{i=1}^{n} p_i k_i = m \cdot \ln 2}.$$

$k_i$ $i \in S$ with prop. $p_i$

$k_i$ will be counted in $\sum_{i \in S} k_i$ with prop $p_i$

$\to 0$ with prop. $1-p_i$

$\to k_i$ with prop. $p_i$

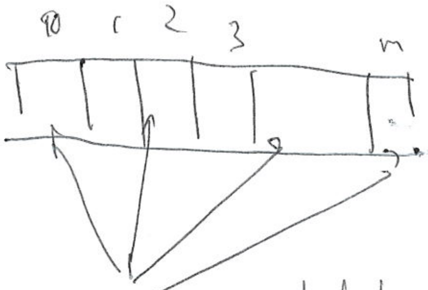
expected value $k_i p_i$

Expected sum value $k_1p_1 + k_2p_2 + k_3p_3 + \dots k_np_n$

$$= \sum_{i=1}^{n} p_i k_i .$$

$$\cancel{\sum_{i \in S}} \boxed{\sum_{i=1}^{n} p_i k_i = m \cdot \ln 2.}$$ constraint.

---




$S_i$ ← check $k_i$ cells. Prop. of false positive from $S_i$ is $\left(\frac{1}{2}\right)^{k_i}$

Minimize $\sum_{i=1}^{n} \left(\frac{1}{2}\right)^{k_i}$.

~~Input~~:

$\cancel{k_i = 0} \to 1$

$k_i = 1 \to \cancel{\frac{1}{2}}$

~~Minimi~~ Maximize $\sum_{i=1}^{n} -\left(\frac{1}{2}\right)^{k_i}$

Maximize $\sum_{i=1}^{n} \left(1 - \left(\frac{1}{2}\right)^{k_i}\right)$.

$k_i = 0 \to 1 - \left(\frac{1}{2}\right)^0 = 0$

$\frac{1}{2} - k_i^{(1)}$

$k_i = 1 \to 1 - \left(\frac{1}{2}\right)^1 = \frac{1}{2}$

$\frac{1}{4} - k_i^{(2)}$

$k_i = 2 \to 1 - \left(\frac{1}{2}\right)^2 = \frac{3}{4}$

$\frac{1}{8} - k_i^{(3)}$

$k_i = 3 \to 1 - \left(\frac{1}{2}\right)^3 = \frac{7}{8}$

$\{0, 1\}$.

$$1 - \left(\frac{1}{2}\right)^{k_i} \to \frac{1}{2} k_i^{(1)} + \frac{1}{4} k_i^{(2)} + \frac{1}{8} k_i^{(3)}$$

$w_i^{(1)} = w_i^{(2)} = w_i^{(3)} = p_i$

$W = m \ln 2$

$h_i^{(1)} = \frac{1}{2}$ $h_i^{(2)} = \frac{1}{4}$

$h_i^{(3)} = \frac{1}{8}$

<u>Input:</u> # possible elements $n$.

$p_1, \dots, p_n$ where $p_i$ is probability that $i \in S$.

<u>Output:</u> $Q \subseteq \left\{ \begin{matrix} k_1^{(1)}, \dots, k_n^{(1)} \\ k_1^{(2)}, \dots, k_n^{(2)} \\ k_1^{(3)}, \dots, k_n^{(3)} \end{matrix} \right\}$

<u>Constraint</u> $$\boxed{\sum_{k_i^{(j)} \in Q} p_i = m \cdot \ln 2} \Rightarrow \sum_{k_i^{(j)} \in Q} w_i^{(j)} \leq W$$

$\leq$

<u>Objective function</u> Maximize $$\sum_{k_i^{(1)} \in Q} \frac{1}{2} + \sum_{k_i^{(2)} \in Q} \frac{1}{4} + \sum_{k_i^{(3)} \in Q} \frac{1}{8}$$

$$= \sum_{k_i^{(1)} \in Q} h_i^{(1)} + \sum_{k_i^{(2)} \in Q} h_i^{(2)} + \sum_{k_i^{(3)} \in Q} h_i^{(3)}$$

$$= \sum_{k_i^{(j)} \in Q} h_i^{(j)}$$

We have knapsack problem!

and we can use $\frac{1}{2}$-approximation algorithm here.