CS341: Algorithms
Lecture 09: Greedy Algorithm III

Yao-Liang Yu

School of Computer Science
University of Waterloo

Feb 02, 2017
Plan for Today

1 Recap

2 Stable Marriage
Plan

1. Recap

2. Stable Marriage
Interval Coloring

Input: A set of $n$ intervals $\{[s_i, f_i]\}$ with start time $s_i$ and finish time $f_i$
Output: Least number of colors such that overlapping intervals are colored differently

Non-optimal greedy algorithm:
- select a maximum number of pairwise disjoint intervals
- color them with a new color
- delete them
- repeat
A lower bound

**Definition** (Depth). Maximum number of intervals that pass same point.

**Theorem.** We need at least as many colors as the depth.

**Corollary.** If an algorithm uses \(d\) colors, then it is optimal.
Algorithm 1: greedyIntervalColor($\mathcal{A}$)

1. $\mathcal{A} \leftarrow \text{sort}(\mathcal{A})$ // by starting time $s_i$
2. $d \leftarrow 1$
3. $\text{color}[1] \leftarrow 1$
4. $\text{finish}[1] \leftarrow f_1$
5. for $i = 2, \ldots, n$ do
6.   $\text{neednewcolor} \leftarrow \text{true}$
7.   $c \leftarrow 1$
8.   while $c \leq d$ do
9.     if $\text{finish}[c] \leq s_i$ then
10.    $\text{color}[i] \leftarrow c$
11.    $\text{finish}[c] \leftarrow f_i$
12.    $\text{neednewcolor} \leftarrow \text{false}$
13.    break
14.   else
15.     $c++$
16.   end if
17.   if $\text{neednewcolor}$ then
18.      $d++$
19.      $\text{color}[i] \leftarrow d$
20.      $\text{finish}[d] \leftarrow f_i$
21. end if
22. return $(d, \text{color})$
Set cover

Set Cover

Input: A set $U = \{1, \ldots, n\}$ and a collection of subsets
$\mathcal{F} = \{A_i \subseteq U : i = 1, \ldots, m\}$
Output: “Smallest” $C \subseteq \mathcal{F}$ such that $\bigcup_{C \in C} C = U$
A greedy approximation algorithm

**Algorithm 2: greedySetCover**(U, \( \mathcal{F} \))

1. \( C \leftarrow \emptyset \)
2. **while** \( U \neq \emptyset \) **do**
3. \[ \text{choose } A \in \mathcal{F} \text{ that maximizes } |A \cap U| \]
4. \( U \leftarrow U \setminus A \)
5. \( C \leftarrow C \cup \{A\} \)
6. **return** \( (C) \)

**greedy** \( \leq \) **opt** \( \log(n) \)
choose $n = 2^{k+1} - 2$

cannot be improved unless $P = NP$ (Dinur & Steurer, 2014)

weighted version, follow the textbook

linear-time implementation (Bar-Yehuda & Even, 1981)
Knapsack

Input: Profits \( P = [p_1, \ldots, p_n] \); weights \( W = [w_1, \ldots, w_n] \); capacity \( C \).

Feasible solution: \( X = [x_1, \ldots, x_n] \) such that \( \sum_i w_i x_i \leq C \), for all \( i \), \( x_i \in \{0, 1\} \) or \( x_i \in \mathbb{N} \) or more relaxedly \( x_i \in [0, 1] \)

Output: A feasible solution \( X \) that maximizes \( \sum_i p_i x_i \).

Possible greedy choices:

- choose the items in decreasing order of profit
- choose the items in increasing order of weight
- choose the items in decreasing order of profit per weight
Algorithm 3: greedyRelaxedKnapsack\((P, W, C)\)

1. \((P, W) \leftarrow \text{sort}(P, W)\) \hspace{1cm} \text{// sort by } p_i/w_i
2. \(i \leftarrow 1\)
3. \(\text{curW} \leftarrow 0\) \hspace{1cm} \text{// current total weight}
4. \textbf{while } \text{curW} < C \text{ and } i \leq n \text{ do}
5. \hspace{1cm} x_i \leftarrow \min\{1, (C - \text{curW})/w_i\}
6. \hspace{1cm} \text{curW} += x_i w_i
7. \hspace{1cm} i++
8. \textbf{return } X

- Clearly, time complexity \(O(n \log n)\)
What if $x_i \in \{0, 1\}$

Would straightforward modification of our greedy algorithm work?

No. In fact, this is an **NP-hard** problem.

Pseudo-polynomial time algorithm using dynamic programming.
Greedy is not too bad

W.l.o.g., assume \( w_i \leq C \) for all \( i \).

- sort by \( p_i/w_i \) decreasingly
- find \( k \) such that \( \sum_{i=1}^{k} w_i \leq C < \sum_{i=1}^{k+1} w_i \)
- return \( \max\{\sum_{i=1}^{k} p_i, p_{k+1}\} \)

Claim: \( \sum_{i=1}^{k} p_i \leq \text{opt} \leq \text{opt}_{\text{relaxed}} < \sum_{i=1}^{k+1} p_i \)

Hence greedy \( \geq \) \( \text{opt}/2 \).

Example: \( P = [2, C], W = [1, C] \)
Plan

1 Recap

2 Stable Marriage
Stable Marriage Problem

Stable Marriage

Input: A set of men $M = [m_1, \ldots, m_n]$ and a set of women $\mathcal{W} = [w_1, \ldots, w_n]$

Each man has a preference ranking (e.g. permutation) of the $n$ women, and vice versa

Output: A (perfect) matching of the men with women so that no pair $(m_i, w_j)$ who are not married to each other but prefer each other to their partners. Such matchings are called stable matching.

- does there exist a stable matching?
- is a stable matching unique?
- how to efficiently find a stable matching?
Algorithm 4: Gale-Shapley \((M, W, \text{pref})\)

1. \(\text{Match} \leftarrow \emptyset\)
2. \textbf{while} \(\exists\) a free man \(m\) \textbf{do}
3. \hspace{1em} let \(w\) be the highest ranked woman in \(m\)’s preference to whom \(m\) has not yet proposed \hspace{1em} // break ties arbitrarily
4. \hspace{1em} \textbf{if} \(w\) is free \textbf{then}
5. \hspace{2em} \text{Match} \leftarrow \text{Match} \cup \{m, w\}
6. \hspace{1em} \textbf{else}
7. \hspace{2em} suppose \((m', w) \in \text{Match}\)
8. \hspace{3em} \textbf{if} \(w\) prefers \(m\) to \(m'\) \textbf{then}
9. \hspace{4em} \text{Match} \leftarrow \text{Match} \setminus \{m', w\} \cup \{m, w\}
10. \hspace{4em} \text{\(m'\) becomes free}
11. \hspace{3em} \textbf{else}
12. \hspace{4em} \text{\(m\) remains free}
13. \textbf{return} \text{Match}
Analysis

- Each woman remains engaged from her first received proposal, and her partner gets better and better (in terms of her preference list).
- Each man proposes to a worse and worse partner (in terms of his preference list).
- Each man proposes at most $n$ times, each woman receives at most $n$ proposals.
- Thus Gale-Shapley terminates after at most $n(n-1) + 1$ iterations.
- A free man can always propose.
- Gale-Shapley returns a stable perfect matching $Match$:
  - If not, $\exists (m, w), (m', w') \in Match$ but $w' >_m w$ and $m >_{w'} m'$
  - $m$ proposed to $w'$ before but was eventually rejected.
  - Thus $w'$ was engaged to someone better than $m$, contradiction.
Why are men proposing?

If each man’s top partner is different, then Gale-Shapley will make every man happy, irrespective of the women’s preferences!

In fact, this is always the case for Gale-Shapley...

For each man $m$, define

\[
\text{valid}(m) := \{w : (m, w) \text{ appears in some stable perfect matching}\}
\]

\[
\text{best}(m) := \{w : w \in \text{valid}(m) \text{ has highest ranking in } m \text{’s preference}\}
\]

\[
\text{Match}^* := \{ (m, \text{best}(m)) \}
\]

Gale-Shapley always outputs $\text{Match}^*$, and $\text{Match}^* = \{ (\text{worst}(w), w) \}$!
Proof

- suppose exist \((\tilde{m}, \tilde{w}) \in Match_{GS}\) and \(\tilde{w} \neq \text{best}(\tilde{m})\)
- recall each man proposes by decreasing order of preference
- some man, say \(m\), was rejected by some \(w \in \text{valid}(m)\)
- when this first happens in GS, \(m\) is rejected by \(w = \text{best}(m)\)
- at the time of this rejection, we infer \(w\) is engaged to \(m'\) (it is possible that \(w\) may reject \(m'\) as well later)
- \(\exists\) a stable \(Match'\) and \(w'\) such that \((m, w), (m', w') \in Match'\)
- when run Gale-Shapley, \(m'\) was not rejected by any \(\text{valid}(m')\) when \(m\) was rejected by \(w\) (since \(m\) is the first poor guy)
- \(m'\) proposes in decreasing order of preference
- \(w' \in \text{valid}(m')\), hence \(w >_{m'} w'\)
- contradiction to the stability of \(Match'\)
Proof cont’

- Suppose \((m, w) \in Match^* \) and \(m \neq \text{worst}(w)\)
- \(\exists\) stable matching \(Match' \ni (m', w)\) and \(m >_w m'\)
- Say \((m, w') \in Match'\) (clearly \(w' \neq w\) since \(w\) is married to \(m'\))
- By previous result \(w = \text{best}(m)\) hence \(w >_m w'\)
- Contradiction to the stability of \(Match'\)
similar algorithm long used for matching medical students with residency hospitals

until 2000s hospital proposed, now you know why :) 

about 10 years ago students started proposing

ancient China: through an intermediate match-maker

Shapley and Roth shared Nobel prize in 2012 (Gale died in 2008)

can you benefit by being dishonest or coordinated? (Dubins & Freedman, 1981)