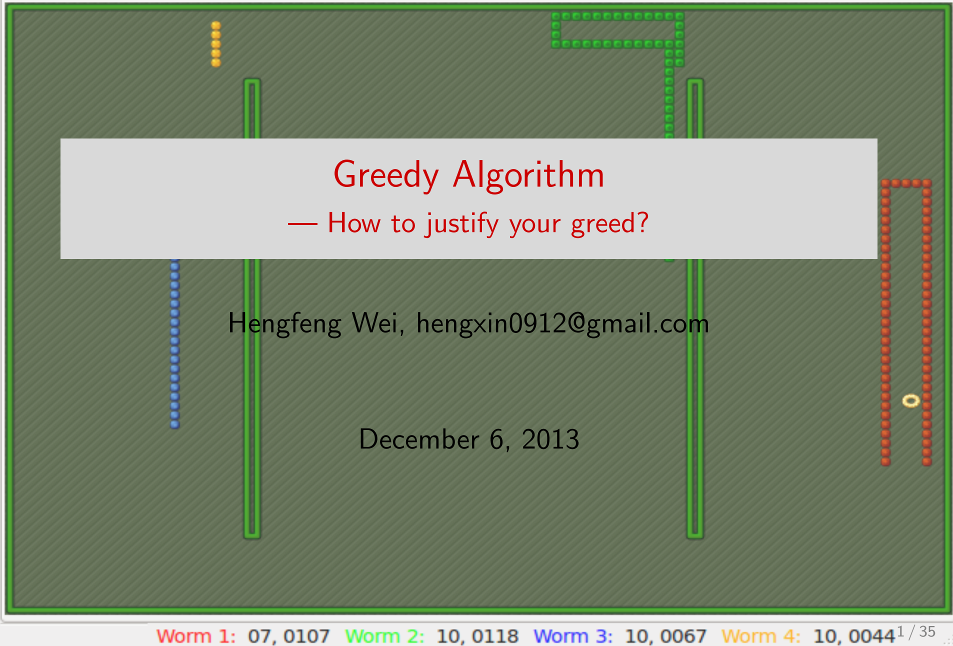
Greedy Algorithm

— How to justify your greed?

Hengfeng Wei, hengxin0912@gmail.com

December 6, 2013



Outline

MST Revisited

Lecture-Selection Problem and Huffman Encoding Problem
- Lecture-Selection Problem
- Huffman Encoding Problem

Waiting Customers Problem

Matroid

MST Revisited

MST revisited:

- to unify Prim and Kruskal algs
- to prove greedy algorithm using *loop invariant*

Prim and Kruskal in common:

- growing the MST one edge at a time
- adding a *safe* edge each time

Generic MST Algorithm

Generic-MST$(G = (V, E), w)$

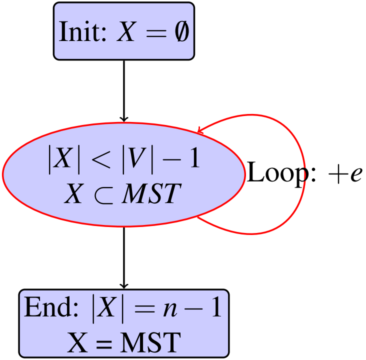
1: $X \leftarrow \emptyset$
2: **while** $|X| < |V| - 1$ **do**
3: find a *safe* edge e
4: $X \leftarrow X \cup e$
5: **end while**
6: **return** X

Correctness Proof

Definition (Loop Invariant)

X is always a subset of some MST of G.

Correctness of MST.



Cut Property

Theorem (Cut Property)

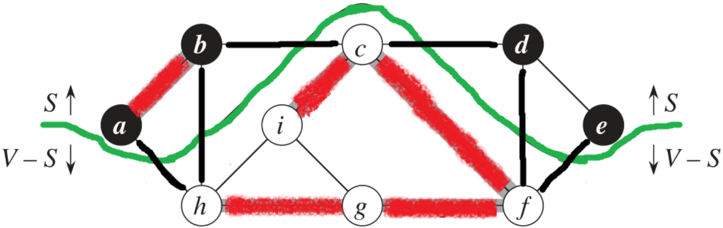
$\boxed{\textit{If}}$ *X is a subset of some MST T of $G = (V, E)$,*

$\boxed{\textit{choice of } e}$:

- *partition V into a cut $(S, V - S)$ such that no edge in X crosses $(S, V - S)$*
- *e is the lightest edge across $(S, V - S)$*

$\boxed{\textit{then,}}$ *$X \cup e$ is a subset of some MST T' of G.*

Cut Property

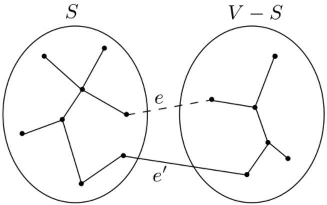


Cut Property

Correctness of Cut Property.

$\boxed{\text{Given}}$ $X \subset T$ (T is some MST), $+e$

- $e \in T$, done.
- $e \notin TS$, $\boxed{\text{Exchange Argument: } T' = T \cup \{e\} - \{e'\}}$



- T' is a tree (connected, $n-1$ edges)
- $w(T') = w(T) - w(e) + w(e') \leq w(T) \Rightarrow w(T') = w(T)$

MST Revisited

Unify Prim & Kruskal algorithms:

Prim: partition between growing tree and other trivial trees

Kruskal: partition between trees in forest

> Summary: Using loop invariant to prove the correctness of greedy algorithms.

Lecture-Selection

Problem (Lecture-Selection)

n lectures $L = \{L_1, L_2, \ldots, L_n\}$ in one day:

- $L_i : (s_i, f_i)$
- *L_i, L_j are compatible (o.w., conflicting) if:*

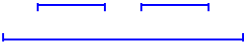
$$s_j > f_i \vee s_i > f_j$$

- $\boxed{\textit{Goal}}$*: To listen to as many mutually compatible lectures as possible.*

Heuristic Lecture-Selection

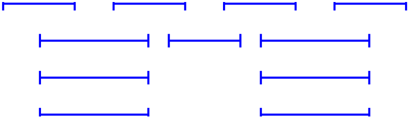
Trial & Error:

- earliest-started lecture first



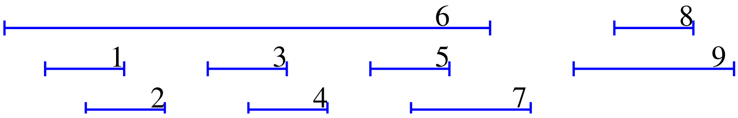
- shortest lecture first

- fewest conflict lecture first



Earliest-Finished Lecture First

Example (Earliest-Finished Lecture First)



It is recursive:

$$L = \{L_1, L_2, \ldots, L_n\} \text{ sorted by } f_i$$

$\boxed{\text{choose } f = L_1}$

$$L' = \{L'_1, L'_2, \ldots\}$$

Correctness Proof

Induction on n:

- B.S.: $n = 1$, trivial.
- I.H.: $< n$ (*strong induction*)
- I.S.: $= n$

Lemma (I.S.)

$$L = \{L_1, L_2, \ldots, L_n\} \text{ sorted by } f_i,$$

$\boxed{\textit{choose } f = L_1}$

$$L' = \{L'_1, L'_2, \ldots\}.$$

$\boxed{\textit{Given (I.H.):}}$ *X' is optimal to L',*

$\boxed{\textit{To prove:}}$ *$X = X' \cup \{f\}$ is optimal to L.*

Correctness Proof

I.S.: Proof by Contradiction.

$L = \{L_1, L_2, \ldots, L_n\}$ sorted by f_i, $\boxed{f = L_1}$ $L' = \{L'_1, L'_2, \ldots\}$.

$$\begin{tikzcd}
(X, L) \arrow[rr, leftrightarrow, "X \leftarrow_{+f} X'", "\text{choose } f = L_1"'] \arrow[d, "\text{suppose } <"'] & & (X', L') \\
(X'', L) \arrow[rr, "\text{choose } f = L_1, X'' \to_{-f} X'''"] & & (X''', L') \arrow[u, "\text{then } <"']
\end{tikzcd}$$

$f \in X''$

optimal substructure

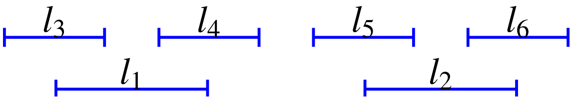
Correctness Proof

Lemma (Optimal Substructure)

Suppose X *is optimal to* L, $f \in X$, L*'s subproblem is* L',

then, X *contains* X' *which is optimal to* L'.

Condition: $\boxed{f \in X \text{ is optimal}}$



- $l_1 \in X = \{l_1, l_2\}$ is not optimal.
- X does not contain the optimal solution to (l_2, l_5, l_6).

Correctness Proof

Lemma (Greedy Choice)

There exists optimal solution containing f to L.

Proof.

Let O be $\boxed{\text{any}}$ optimal solution to L.

- $f \in O$, done.
- $f \notin O$.
  - $g \leftarrow$ the earliest-finished lecture in O
  - $\boxed{\text{Exchange Argument: } O' = O - \{g\} \cup \{f\}}$

$\square$

Lecture-Selection

Prove the correctness of greedy algorithms:

- apply $\boxed{\text{Greedy Choice}}$ and $\boxed{\text{Optimal Substructure}}$ in mathematical induction;
- apply $\boxed{\text{Exchange Argument}}$ in $\boxed{\text{Greedy Choice}}$

Huffman Encoding

Problem (Huffman 01 Encoding (1))

- *characters* $C[1 \ldots n]$*; frequencies* $F[1 \ldots n]$
- *fixed length code*
- *variable length code*
  - *Morse code (e.g., SOS)*
  - *prefix code: no code is a prefix of some other code*

This sentence contains three a's, three c's, two d's, twenty-six e's, five f's, three g's, eight h's, thirteen i's, two l's, sixteen n's, nine o's, six r's, twenty-seven s's, twenty-two t's, two u's, five v's, eight w's, four x's, five y's, and only one z.

a	b	c	d	e	f
45	13	12	16	9	5
000	001	010	011	100	101
0	101	100	111	1101	1100

Huffman Encoding

Problem (Huffman 01 Encoding (2))

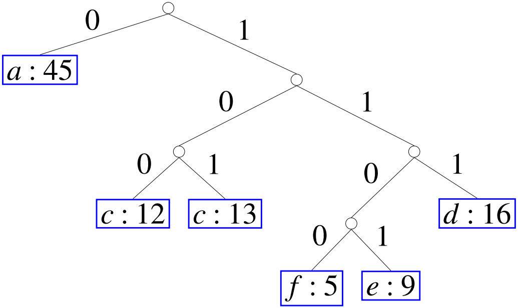
E : binary prefix code

$$L(E) = \sum_{c \in C} f_c \cdot l_E(c), \qquad L = \min_E L(E)$$

Binary Prefix Code

Representation of binary prefix code by *full* binary tree

a	0
b	101
c	100
d	111
e	1101
f	1100



$$L(T) = \sum_{c \in C} f(c) \cdot d_T(c) \qquad L = \min_T L(T)$$

Huffman Encoding Algorithm

It is recursive:

$$C[1 \dots n], F[1 \dots n],$$

$\boxed{f_{n+1} = f_1 + f_2 \text{ as lowest siblings}}$

$$C'[3 \dots n+1], F'[3, \dots n+1]$$

Correctness Proof

Induction on $n = |C|$:

- B.S.: $n = 1, 2$, trivial.
- I.H.: $< n$
- I.S.: $= n$

Lemma (I.S.)

$C[1 \ldots n], F[1 \ldots n]$, $\boxed{f_{n+1} = f_1 + f_2}$, $C'[3 \ldots n+1], F'[3, \ldots n+1]$.

$\boxed{\textit{Given (I.H.):}}$ *T' is optimal for C', F',*

$\boxed{\textit{To prove:}}$ *$T' + \{f_1, f_2\}$ is optimal for C, F.*

Correctness Proof

I.S.: Proof by Contradiction.

$C[1 \dots n], F[1 \dots n]$, $\boxed{f_{n+1} = f_1 + f_2}$, $C'[3 \dots n+1], F'[3, \dots n+1]$.

$$\begin{tikzcd}[column sep=huge, row sep=huge]
(T, C, F) \arrow[d, "\substack{\text{suppose} \\ <}"'] & (T', C', F') \arrow[l, "\text{choose } f = f_1 + f_2"', "\text{unfold } T \xleftarrow{f_1, f_2} T'"] \\
(T'', C, F) \arrow[r, "\text{merge } f_{n+1} = f_1 + f_2,\ T'' \to T'''"] & (T''', C', F') \arrow[u, "\substack{\text{then} \\ <}"']
\end{tikzcd}$$

f_1, f_2 as lowest siblings

optimal substructure

Correctness Proof

Merge ($T \to T'$; $T'' \to T'''$):

$$\begin{aligned} L(T) &= \sum_{i=1}^{n} f(i) d_T(c_i) \\ &= L(T') - f_{n+1} \cdot d_{T'}(c_{n+1}) + f_1 \cdot d_T(c_1) + f_1 \cdot d_T(c_1) \\ &= L(T') - (f_1 + f_2) \cdot (d_T(c_1) - 1) + (f_1 + f_2) \cdot d_T(c_1) \\ &= L(T') + f_1 + f_2. \end{aligned}$$

$$L(T'') = L(T''') + f_1 + f_2$$

$$L(T''') = L(T'') - f_1 - f_2 < L(T) - f_1 - f_2 = L(T')$$

Correctness Proof

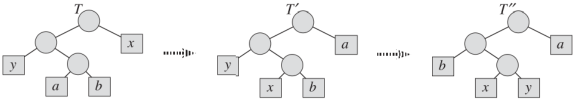
Lemma (Greedy Choice)

$\boxed{\text{Let}}$ x, y be two least frequent characters of (C, F),
There $\boxed{\text{exists}}$ optimal code tree with x, y as lowest siblings.

Proof.

Let T be $\boxed{\text{any}}$ optimal code tree.

- with x, y as lowest siblings, done.
- o.w., $\boxed{\text{Exchange Argument: } T \Rightarrow T' \Rightarrow T''(a \Leftrightarrow x, b \Leftrightarrow y)}$



$$f(x) \le f(a), f(y) \le f(b) \Rightarrow L(T'') \le L(T') \le L(T)$$

T'' is optimal to (C, F) with x, y as lowest siblings.

Huffman Encoding

Prove the correctness of greedy algorithms:

- apply $\boxed{\text{Greedy Choice}}$ and $\boxed{\text{Optimal Substructure}}$ in mathematical induction;
- apply $\boxed{\text{Exchange Argument}}$ in $\boxed{\text{Greedy Choice}}$

Waiting Customers

Problem (Waiting Customers (P10))

- *n waiting customers* $t_1, t_2, \dots, t_k \dots t_n$
- *for* k^{th}: $T(k) = \sum_{i=1}^{k} t_i$
- *to minimize* $T = \sum_{k=1}^{n} \sum_{i=1}^{k} t_i$

Claim: T is minimized iff $t_1, \dots, t_n$ are in non-decreasing order.

Prove by Contradiction:

$\boxed{\text{Otherwise,}}$ there are, in T two neighbouring pair $t_i > t_{i+1}$ and t_i is before t_{i+1},

$\boxed{\text{Exchange Argument: swap } t_i \Leftrightarrow t_{i+1} \text{ to get } T'}$.

$$T' - T = (t_{i+1} + (t_i + t_{i+1})) - (t_i + (t_{i+1} + t_i)) = t_{i+1} - t_i < 0.$$

Matroid

Matroid: characterize mathematically the problems which can be solved by greedy algorithm in many cases.

With Matroid:

- cast your problem in terms of Matroid language
- Matroid has the $\boxed{\text{Greedy Choice Property}}$ and the $\boxed{\text{Optimal Substructure Property}}$
- Matroid problem can be solved by greedy algorithm mechanically

[Section 16.4 @CLRS]

Summary

Greedy algorithm — How to justify your greediness?

- loop invariant
  - MST, Dijkstra
- greedy choice + optimal substructure
  - lecture-selection, Huffman encoding
- exchange argument
  - waiting customers
- Matroid theory