Scheduling

L10

CS4800 F16

abhi shelat
<table>
<thead>
<tr>
<th>Course Code</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>sy3333</td>
<td>2</td>
<td>3.25</td>
</tr>
<tr>
<td>en1612</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>ma1231</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>cs4102</td>
<td>3.5</td>
<td>4.75</td>
</tr>
<tr>
<td>cs4800</td>
<td>4</td>
<td>5.25</td>
</tr>
<tr>
<td>cs6051</td>
<td>4.5</td>
<td>6</td>
</tr>
<tr>
<td>sy3100</td>
<td>5</td>
<td>6.5</td>
</tr>
<tr>
<td>cs1000</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>
problem statement

\[(a_1, \ldots, a_n)\]
\[(s_1, s_2, \ldots, s_n)\]
\[(f_1, f_2, \ldots, f_n) \text{ (sorted)} \quad s_i < f_i\]

find largest subset of activities \(C=\{a_i\}\) such that
\[(\text{compatible})\]
problem statement

\[(a_1, \ldots, a_n)\]
\[(s_1, s_2, \ldots, s_n)\]
\[(f_1, f_2, \ldots, f_n)\] (sorted) \[s_i < f_i\]

find largest subset of activities \(C=\{a_i\}\) such that

(compatible)

\[a_i, a_j \in C, i < j\]

\[f_i \leq s_j\]
problem statement

\[(a_1, \ldots, a_n)\]
\[(s_1, s_2, \ldots, s_n)\]

\[(f_1, f_2, \ldots, f_n) \text{ (sorted)} \quad s_i < f_i\]
dynamic programming
dynamic programming

\[ \text{BEST}_{f_n} = \max \text{ BEST}_{s_n} + 1 \quad \text{in: } a_n \]

\[ \text{BEST}_{e_t} \quad \text{out: } a_n \]
greedy solution:

**Definition:**

\[ \text{SOLTN}_{i,j} \]
greedy solution:

\[ s_1 \rightarrow f_1 \rightarrow f_2 \]

\[ \text{goal: } \text{SOLTN}_{0,2n} \]
greedy solution:

claim: the first action to finish in $e[i,j]$ is always part of some $\text{SOLTN}_{i,j}$
claim: the first action to finish in $e[i,j]$ is always part of some $\text{SOLTN}_{i,j}$

proof:
**greedy solution:**

**algorithm:**

1. find first event to finish. add to solution.
2. remove conflicting events.
3. continue.
greedy solution:

**algorithm:** find first event to finish. add to solution. remove conflicting events. continue.
greedy solution:

**algorithm:**
find first event to finish. add to solution.
remove conflicting events.
continue.
greedy solution:

algorithm:

- Find first event to finish. Add to solution.
- Remove conflicting events.
- Continue.
greedy solution:

**algorithm:**
find first event to finish. add to solution.
remove conflicting events.
continue.
greedy solution:

algorithm: find first event to finish. add to solution. remove conflicting events. continue.
greedy solution:

algorithm: find first event to finish. add to solution. remove conflicting events. continue.
algorithm: find first event to finish. add to solution.
remove conflicting events.
continue.

\((f_1, f_2, \ldots, f_n)\text{ (sorted)}\quad s_i < f_i\)
caching L10 CS4800
cache hit

Cache

CPU

load r2, addr a
store r4, addr b

main memory
question:
problem statement

input:

output:

cache is
problem statement

input: \( K \), the size of the cache
\( d_1, d_2, \ldots, d_m \) memory accesses

output: schedule for that cache that minimizes # of cache misses while satisfying requests

_cache is fully associative, line size is 1_
contrast with reality
Belady evict rule
example

cache

a
b
c

a b c d a d e a d b a e c e a
example

cache

a
b
c

a
b
d

da
d
e
da
d
b
a
e
c
ea

a
b
c
d

da
d
b
a
e
c
ea
example

cache

a b c d a d e a d b a e c e a
example
example

cache

a b c d a d e a d b a e c e a

a a a a a a
b b b b e e
a d d d b b
a c c e e

a
b
a
b
a
c
b
e
e
Surprising theorem
Schedule for access pattern d₁,d₂,...,dₙ:

Reduced schedule:
Exchange lemma
Exchange Lemma:

Let \( S \) be a reduced schedule that agrees with \( S_{ff} \) on the first \( j \) items. There exists a reduced schedule \( S' \) that agrees with \( S_{ff} \) on the first \( j+1 \) items and has the same or fewer # misses as \( S \).
$S^*$

$S^f_f$
Proof of Lemma

Let $S$ be a reduced sched that agrees with $S_{ff}$ on the first $j$ items. There exists a reduced sched $S'$ that agrees with $S_{ff}$ on the first $j+1$ items and has the same or fewer #misses as $S$. 
Proof of lemma

State of the cache after J operations under the two schedules.

\[
\begin{align*}
S & \quad e \quad f \\
S_{ff} & \quad e \quad f
\end{align*}
\]

easy case 1

easy case 2
Proof of lemma

case 3
Timeline

S_{ff}

S'
Proof of lemma

Let access t
Proof of lemma

what if t=e?
Proof of lemma

what if $t=f$?
Proof of lemma

what if $t$ is neither $e$ nor $f$?
Let $S$ be a reduced sched that agrees with $S_{ff}$ on the first $j$ items. There exists a reduced sched $S'$ that agrees with $S_{ff}$ on the first $j+1$ items and has the same or fewer #misses as $S$. 

What have we shown
Let $S$ be a reduced sched that agrees with $S_{ff}$ on the first $j$ items. There exists a reduced sched $S'$ that agrees with $S_{ff}$ on the first $j+1$ items and has the same or fewer #misses as $S$. 
Huffman

L10

CS4800
MOSCOW — President Vladimir V. Putin’s typically theatrical order to withdraw the bulk of Russian forces from Syria, a process that the Defense Ministry said it began on Tuesday, seemingly caught Washington, Damascus and everybody in between off guard — just the way the Russian leader likes it.

By all accounts, Mr. Putin delights at creating surprises, reinforcing Russia’s newfound image as a sovereign, global heavyweight and keeping him at the center of world events.
MOSCOW — President Vladimir V. Putin’s typically theatrical order to withdraw the bulk of Russian forces from Syria, a process that the Defense Ministry said it began on Tuesday, seemingly caught Washington, Damascus and everybody in between off guard — just the way the Russian leader likes it.

By all accounts, Mr. Putin delights at creating surprises, reinforcing Russia’s newfound image as a sovereign, global heavyweight and keeping him at the center of world events.
\( c \in C \quad f_c \quad T \)

\begin{align*}
  e & : 235 \\
  i & : 200 \\
  o & : 170 \\
  u & : 87 \\
  p & : 78 \\
  g & : 47 \\
  b & : 40 \\
  f & : 24 \\
\end{align*}

881
<table>
<thead>
<tr>
<th>$c \in C$</th>
<th>$f_c$</th>
<th>$T$</th>
<th>$l_c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
<td>235</td>
<td>000</td>
<td>3</td>
</tr>
<tr>
<td>i</td>
<td>200</td>
<td>001</td>
<td>3</td>
</tr>
<tr>
<td>o</td>
<td>170</td>
<td>010</td>
<td>3</td>
</tr>
<tr>
<td>u</td>
<td>87</td>
<td>011</td>
<td>3</td>
</tr>
<tr>
<td>p</td>
<td>78</td>
<td>100</td>
<td>3</td>
</tr>
<tr>
<td>g</td>
<td>47</td>
<td>101</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>40</td>
<td>110</td>
<td>3</td>
</tr>
<tr>
<td>f</td>
<td>24</td>
<td>111</td>
<td>3</td>
</tr>
</tbody>
</table>

881
def: cost of an encoding

\[ B(T, \{ f_c \}) = \sum_{c \in C} f_c \cdot \ell_c \]

<table>
<thead>
<tr>
<th>( c \in C )</th>
<th>( f_c )</th>
<th>( T )</th>
<th>( \ell_c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>e:</td>
<td>235</td>
<td>000</td>
<td>3</td>
</tr>
<tr>
<td>i:</td>
<td>200</td>
<td>001</td>
<td>3</td>
</tr>
<tr>
<td>o:</td>
<td>170</td>
<td>010</td>
<td>3</td>
</tr>
<tr>
<td>u:</td>
<td>87</td>
<td>011</td>
<td>3</td>
</tr>
<tr>
<td>p:</td>
<td>78</td>
<td>100</td>
<td>3</td>
</tr>
<tr>
<td>g:</td>
<td>47</td>
<td>101</td>
<td>3</td>
</tr>
<tr>
<td>b:</td>
<td>40</td>
<td>110</td>
<td>3</td>
</tr>
<tr>
<td>f:</td>
<td>24</td>
<td>111</td>
<td>3</td>
</tr>
</tbody>
</table>

881
morse code

image http://en.wikipedia.org/wiki/Morse_code
morse code

International Morse Code
- 1 dash = 3 dots.
- The space between parts of the same letter = 1 dot.
- The space between letters = 3 dots.
- The space between words = 7 dots.

A ▲ ▲ ▲ ▲
B ▲ ▲ ▲ ▲ ▲ ▲ ▲
C ▲ ▲ ▲ ▲ ▲ ▲ ♦
D ▲ ▲ ▲ ♦
E ▲ ♦
F ♦ ♦ ♦ ♦ ♦
G ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦
H ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦
I ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦
J ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦
K ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦
L ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦
M ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦
N ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦
O ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦
P ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦
Q ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦
R ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦
S ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦
T ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦
U ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦
V ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦
W ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦
X ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦
Y ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦
Z ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦

.--

3 dots
def: prefix-free code
def: prefix-free code

\[ \forall x, y \in C, x \neq y \implies \text{CODE}(x) \text{ not a prefix of } \text{CODE}(y) \]
def: prefix code

∀x, y ∈ C, x ≠ y ⟹ CODE(x) not a prefix of CODE(y)

e: 235 0
i: 200 10
o: 170 110
u: 87 1110
p: 78 11110
g: 47 111110
b: 40 1111110
f: 24 11111110
decoding a prefix code

e: 235  0
i: 200  10
o: 170  110
u: 87   1110
p: 78   11110
G: 47   111110
b: 40   1111110
f: 24   11111110

111111010111110
code to binary tree

e: 235  0
i: 200  10
o: 170  110
u: 87   1110
p: 78   11110
g: 47   111110
b: 40   1111110
f: 24   11111110

11111101010111110
prefix code

binary tree
use tree to encode

c ∈ C   f_c   T   \ell_c
---   ----   ---   ---
e: 235 00   2
i: 200 01   2
o: 170 10   2
u:  87 110  3
p:  78 111  3
given the goal
given the character frequencies \( \{ f_c \} \) \( c \in C \) (all frequencies are > 0)

produce a prefix code \( T \) with smallest cost

\[
\min_T B(T, \{ f_c \})
\]
lemma: optimal tree must be full.
divide & conquer?
counter-example

e: 32
i: 25
o: 20
u: 18
p: 5
e: 235 01
i: 200 11
o: 170 10
u: 87 0011
p: 78 0010
g: 47 0000
b: 40 00011
f: 24 00010
objective
exchange argument

lemma:
Lemma. Let \( x, y \in C \) be characters with smallest frequencies \( f_x, f_y \). There exists an optimal prefix code \( T'' \) for \( C \) in which \( x, y \) are siblings. That is, the codes for \( x, y \) have the same length and only differ in the last bit.
Let $x, y \in C$ be characters with smallest frequencies $f_x, f_y$. There exists an optimal prefix code $T''$ for $C$ in which $x, y$ are siblings. That is, the codes for $x, y$ have the same length and only differ in the last bit.
exchange argument

lemma:

Let $x, y \in C$ be characters with smallest frequencies $f_x, f_y$. There exists an optimal prefix code $T''$ for $C$ in which $x, y$ are siblings. That is, the codes for $x, y$ have the same length and only differ in the last bit.

proof:
Let $x, y \in C$ be characters with smallest frequencies $f_x, f_y$. There exists an optimal prefix code $T''$ for $C$ in which $x, y$ are siblings. That is, the codes for $x, y$ have the same length and only differ in the last bit.
Exchange Argument

Lemma. Let \( x, y \in C \) be characters with smallest frequencies \( f_x, f_y \). There exists an optimal prefix code \( T'' \) for \( C \) in which \( x, y \) are siblings. That is, the codes for \( x, y \) have the same length and only differ in the last bit.

\[
\begin{align*}
f_a &\leq f_b \\
f_x &\leq f_y \\
f_y &\leq f_b
\end{align*}
\]
The diagram illustrates a transformation from tree $T$ to tree $T'$. The process involves changing the positions of nodes $x$, $b$, and $y$.
\[ B(T) = \sum_c f_c \ell_c + f_x \ell_x + f_a \ell_a \quad B(T') = \sum_c f_c' \ell_c' + f_x \ell_x' + f_a \ell_a' \]

\[ B(T) - B(T') \geq 0 \]
exchange argument

\[ B(T') - B(T'') \geq 0 \]
\[ B(T) - B(T') \geq 0 \quad \text{and} \quad B(T') - B(T'') \geq 0 \]

\(T''\) is also optimal
Let $x, y \in C$ be characters with smallest frequencies $f_x, f_y$. There exists an optimal prefix code $T''$ for $C$ in which $x, y$ are siblings. That is, the codes for $x, y$ have the same length and only differ in the last bit.
optimal sub-structure

\[ f_c \quad \square \quad \square \quad \square \quad \square \quad \square \quad \square \quad f_x \quad f_y \]
optimal sub-structure

\( f_c \) problem of size \( n \)

\( f_x \) \( f_y \)

\( f_y \) problem of size \( n-1 \)

\( f_z \)
Lemma:
optimal sub-structure

Lemma: The optimal solution for $T$ consists of computing an optimal solution for $T'$ and replacing the left $z$ with a node having children $x, y$. 
\[ B(T') = B(T) - f_x - f_y \]
Suppose $T$ is not optimal
Suppose $T$ is not optimal.

$B(U) < B(T)$
Suppose $T$ is not optimal

$B(U) < B(T)$
Suppose $T$ is not optimal

$B(U) < B(T)$

$B(U') = B(U) - f_x - f_y$

$< B(t) - f_x - f_y$

But this implies that $B(T')$ was not optimal.
therefore
summary of argument