Greedy Alg:
Caching,
CACHING
CACHE HIT

Cache
load r2, addr a
main memory
store r4, addr b

CPU
QUESTION:

How to manage the cache??

Best-case scenario in which the pattern of all memory accesses is known a priori.
PROBLEM STATEMENT

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

output:  \( \text{min } \# \text{ of cache misses} \)

        cache is fully associative, line size is 1
BELADY EVICT RULE

"farthest in the future" or FF

If you must evict, evict the element that is accessed the farthest in the future.
EXAMPLE

cache

a
b
c

da b c d a d e a d b a e c e a
EXAMPLE

cache

```
ab
c
```

```
a
b
d
```

```
a b c d a d e a d b a e c e a
```
EXAMPLE

cache

```plaintext
  a
  b
  c
  d
  a
  b
  e
  d
```

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

cache

```
  a   a   a   a
  b   b   e   e
  c   d   d   b
  a b c d a d e a d b a e c e a
```
EXAMPLE

Space Fill Schedule (SFS)
operations to
the cache
that follow SFS

Another Schedule
that has the
same # of misses.
The schedule in which we evict the item that is accessed farthest-in-the-future, ie, \textit{S}ff, is optimal.
Schedule for access pattern d₁, d₂, ..., dₙ:

Operation on the cache at each access
- "nop" or "evict x for y"

Reduced schedule:

Schedule in which "evict x for y" only occurs when the access is dirty

For any schedule S, \( \text{misses}(\text{Reduced}(S)) \leq \text{misses}(S) \)
EXCHANGE LEMMA

If $S$ is a reduced schedule that agrees w/ $S_{ff}$ on the first $j$ ops, then $\exists$ a reduced schedule $S'$ that agrees w/ $S_{ff}$ on $|S'|$ operations, and

$$\text{misses}(S') \leq \text{misses}(S).$$
Exchange Lemma:

Let $S$ be a reduced sched that agrees with $S_{ff}$ on $j$ items. There exists a reduced sched $S'$ that agrees on $j+1$ items and has the same or fewer # of misses as $S$. 
Why do we care about this lemma? 

\[
\text{misses}(S_{ff}) \leq \text{misses}(S_{opt})
\]
Thm: Let $S$ be a reduced sched that agrees with $S_{ff}$ on $j$ items. There exists a reduced sched $S'$ that agrees on $j+1$ items and has the same # of misses as $S$.

Proof:

Since $S$ and $S_{ff}$ agree on the first $j$ operations, both schedules produce the same cache state.

Let $d$ be the address accessed at time $j+1$. 
State of the cache after \( J \) operations under the two schedules.

\[
\begin{array}{c}
S \\
\text{e} \quad \text{f} \\
S_{ff} \\
\text{e} \quad \text{f}
\end{array}
\]

easy case 1
\[d \in \text{cache}\]

easy case 2
\[d \notin \text{cache} \text{ but both } S \text{ and } S_{ff} \text{ “evict } e \text{ for } d.\]

\[\Rightarrow \text{Both } S \text{ and } S_{ff} \text{ also agree on the first J+1 operations.}\]

So set \( S' = S \). Done.
case 3: d & cache. Sevicts e but Sff evicts f.

so how can we construct $S'$ in this case? ?

$S'$ must do what Sff does and

"evict f for d"
THE CONSTRUCTION OF $S'$

$S_{ff}$

$S'$

$S$

first operation in which $e$ or $f$ is involved after $\mathcal{F}$.
Let access \( t \) be the first access involving \( e \) or \( f \) after \( j+1 \).

1. Set \( S \) & \( S' \) will be the same as this point except for \( 2df3 \cup \{ e, d \} \).

Cases to consider:

- \( t \) accesses \( e \).
- \( t \) accesses \( f \).
- \( t \) accesses \( g + e + f \).
what if t=e?

1. S must evict some element to bring in e.
   - if S evicts f, then $S + S'$ agree.
   - if S "evicts h for f" ⇒ make $S'$ "evict h for f"

$S \quad d \quad e \quad f$

$S' \quad h \quad l \quad e \quad d$

misses($S'$) < misses($S$)

$S \quad e \quad l \quad d \quad f$

$S' \quad l \quad f \quad e \quad d$

misses($S'$) = misses($S$)
what if $t=f$ ?

IMPOSSIBLE!

$S_0$ evicted $f$ instead of $e$. This means that

$f$ had to be accessed after $e \rightarrow$ farther in the future
what if \( t \) is neither \( e \) nor \( f \) ?

\[ \Rightarrow S \text{'s operation must be } \text{"evict } f \text{ for } g \text{"} \]

\[ S = \text{[T]} \quad \text{d} \quad \text{g} \]

\[ S' = \text{[T]} \quad \text{e} \quad \text{g} \quad \text{d} \]

\[ S' : \text{"evict } e \text{ for } g \text{"} \]

\( \text{misses } (S) = \text{misses } (S') \)

Finally, we set

\[ S' = \text{Reduce}(S') \text{ and conclude thm.} \]
WHAT HAVE WE SHOWN

$S_f$

$S'$

$S$

\[ \text{misses}(S') \leq \text{misses}(S) \]
$S^*$

$S_{ff}$
Huffman Coding
In testimony before the committee, Mr. Lew stressed that the Treasury Department would run out of “extraordinary measures” to free up cash in a matter of days. At that point, the country’s bills might overwhelm its cash on hand plus any receipts from taxes or other sources, leading to an unprecedented default. Mr. Lew said that Treasury had no workarounds to avoid breaching the debt ceiling. “There is no plan other than raising the debt limit,” he said. “The legal issues, even regarding interest and principal on the debt, are complicated.”
In testimony before the committee, Mr. Lew stressed that the Treasury Department would run out of “extraordinary measures” to free up cash in a matter of days. At that point, the country’s bills might overwhelm its cash on hand plus any receipts from taxes or other sources, leading to an unprecedented default. Mr. Lew said that Treasury had no workarounds to avoid breaching the debt ceiling. “There is no plan other than raising the debt limit,” he said. “The legal issues, even regarding interest and principal on the debt, are complicated.”
\[ c \in C \quad f_c \quad T \]

\[ \begin{array}{l}
e: \ 235 \\
i: \ 200 \\
o: \ 170 \\
u: \ 87 \\
p: \ 78 \\
g: \ 47 \\
b: \ 40 \\
f: \ 24 \\
\end{array} \]

\[ 881 \]
\[ c \in C \quad f_c \quad T_c \quad l_c \]

<table>
<thead>
<tr>
<th>c</th>
<th>( f_c )</th>
<th>( T_c )</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>

\[ \text{cost of sending an 881 character msg would be } 881 \cdot 3 = 2643 \]

Q: Can we do better?
DEF: COST OF AN ENCODING

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

c \in C  |  f_c  |  T  |  l_c  \\
--- | --- | --- | --- 
 e: 235  | 000  | 3  
 i: 200  | 001  | 3  
 o: 170  | 010  | 3  
 u: 87   | 011  | 3  
 p: 78   | 100  | 3  
 g: 47   | 101  | 3  
 b: 40   | 110  | 3  
 f: 24   | 111  | 3  

881

\[ l_c = 3 \text{ for every } c \in C \]
MORSE CODE
DEF: PREFIX-FREE CODE

for every $x, y \in C \quad x \neq y$

$c(x)$ is not a prefix of $c(y)$
DEF: PREFIX-FREE CODE

∀x, y ∈ C, x ≠ y → CODE(x) not a prefix of CODE(y)
**DEF: PREFIX CODE**

\( \forall x, y \in C, x \neq y \implies \text{CODE}(x) \text{ not a prefix of CODE}(y) \)

<table>
<thead>
<tr>
<th>Letter</th>
<th>Code</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
<td>235</td>
<td>(0)</td>
</tr>
<tr>
<td>i</td>
<td>200</td>
<td>(10)</td>
</tr>
<tr>
<td>o</td>
<td>170</td>
<td>(110)</td>
</tr>
<tr>
<td>u</td>
<td>87</td>
<td>(1110)</td>
</tr>
<tr>
<td>p</td>
<td>78</td>
<td>(11110)</td>
</tr>
<tr>
<td>g</td>
<td>47</td>
<td>(111110)</td>
</tr>
<tr>
<td>b</td>
<td>40</td>
<td>(1111110)</td>
</tr>
<tr>
<td>f</td>
<td>24</td>
<td>(11111110)</td>
</tr>
</tbody>
</table>
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

B i c
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

prefix-free code →

111111010111110

B ⊥ C
PREFIX CODE

BINARY TREE
USE TREE TO ENCODE MESSAGES
GOAL

GIVEN THE

frequencies for an alphabet $\mathcal{C}$, given the

prefix-free binary tree

compute the (encoding) $T$ such that

minimizes $\sum_{c \in \mathcal{C}} B(T, \mathcal{C}, f_c)$
GOAL

GIVEN THE CHARACTER FREQUENCIES

\[ \{ f_c \} c \in C \]

PRODUCE A PREFIX CODE T WITH SMALLEST COST

\[ \min_T B(T, \{ f_c \}) \]
PROPERTY

LEMMA: OPTIMAL TREE MUST BE FULL.
DIVIDE & CONQUER?
e: 235 01
i: 200 11
o: 170 10
u: 87 0011
p: 78 0010
g: 47 0000
b: 40 00011
f: 24 00010
e: 235 01
i: 200 11
o: 170 10
u: 87 0011
p: 78 0010
g: 47 0000
b: 40 00011
f: 24 00010

2378

versus

2643
OBJECTIVE
EXCHANGE ARGUMENT

LEMMA:
**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.
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.
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:
**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.

![Diagram](image-url)

**FIRST STEP**
**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.

\[
\begin{align*}
f_a &\leq f_b \\
f_x &\leq f_a \\
f_x &\leq f_y \\
f_y &\leq f_b
\end{align*}
\]
\( f_x \leq f_a \)

\[
B(T) = \sum_c f_c \ell_c + f_x \ell_x + f_a \ell_a \\
B(T') = \sum_c f_c' \ell'_c + f_x' \ell'_x + f_a' \ell'_a
\]

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

\( T'' \) is also optimal
**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.
**OPTIMAL SUB-STRUCTURE**

\[ f_c \]

<table>
<thead>
<tr>
<th>235</th>
<th>200</th>
<th>170</th>
<th>87</th>
<th>78</th>
<th>47</th>
<th>( f_x )</th>
<th>( f_y )</th>
</tr>
</thead>
</table>

\( f_x \) and \( f_y \) represent the values of the function in specific directions.
# OPTIMAL SUB-STRUCTURE

<table>
<thead>
<tr>
<th>n</th>
<th>235</th>
<th>200</th>
<th>170</th>
<th>87</th>
<th>78</th>
<th>47</th>
<th>40</th>
<th>24</th>
</tr>
</thead>
</table>

**Problem of size \( n \)**

<table>
<thead>
<tr>
<th>n-1</th>
<th>235</th>
<th>200</th>
<th>170</th>
<th>87</th>
<th>78</th>
<th>47</th>
<th>64</th>
</tr>
</thead>
</table>

**Problem of size \( n-1 \)**
Lemma:
**OPTIMAL SUB-STRUCTURE**

\[
f_c \begin{bmatrix} 235 & 200 & 170 & 87 & 78 & 47 & f_x \ 40 & f_y \end{bmatrix} \]

**Problem of size n**

\[
f_{c'} \begin{bmatrix} 235 & 200 & 170 & 87 & 78 & 47 & 64 \ f_z \end{bmatrix} \]

**Problem of size n-1**

**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

$B(U) < B(T)$

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

$< B(t) - fx - fy$

But this implies that $B(T')$ was not optimal.
THEREFORE

\[ T' \]

\[ x \quad y \]
SUMMARY OF ARGUMENT