\[ A_i \cap B_i = M \leq \# \text{ people in a district} \]

\[ A_i = \# \text{ of people that vote for A in precinct } i \]
GERRYMANDER PROBLEM

given: \( M, A, A_2, ... A_n \) \( n \) is even

output: 2 districts \( D_1, D_2 \). \( |D_1| = |D_2| \) i.e. same # of precincts

\[ A(D_1) > \frac{M \cdot n}{4} \] i.e. party A has a majority

\[ A(D_2) > \frac{M \cdot n}{4} \] in both districts.
GERRYMANDER PROBLEM

given:

\[ m \ A_1, A_2, \ldots, A_n \]

output:

\[ D_1, D_2 \]

such that

\[ |D_1| = |D_2| \]

\[ A(D_1) > \frac{mn}{4} \]

\[ A(D_2) > \frac{mn}{4} \]

or “failure” if no such solution is possible
GERRYMANDER

imagine very last precinct and how it is assigned:
$S_{j,k,x,y} = \text{true or false variable}$

$\text{TRUE if } j \text{ an assignment of the first } j \text{ precincts s.t.}$

$|D_i| = n$

$A(D_i) = x$

$A(D_c) = y$. 
GERRYMANDER

\[ S_{j,k,x,y} = \text{there is a split of first } j \text{ precincts in which } |D_1|=k \text{ and} \]
\[ \text{x people in } D_1 \text{ vote A} \]
\[ \text{y people in } D_2 \text{ vote A} \]

\[ S_{j,k,x_1,y} = S_{j-1,k-1,x-A_j,y} \text{ or } S_{j-1,k,x_1,y-A_j} \]
Brute force

\[ \left( \frac{n}{2} \right)^n - 2^{n/2} \]
\[
S_{j,k,x,y} = S_{j-1,k-1,x-A_j,y} \lor S_{j-1,k,x,y-A_j}
\]

**GERRYMANDER(P,A,m)**

\[C_{j,x,y} = 1 \text{ or } 2\]

initialize array \(S[0,o,o,o]\)

for \(j=1 \text{ to } n\)

\[\theta(n)\]

for \(k=1 \text{ to } n/2\)

\[\theta(n)\]

for \(x=1 \text{ to } M\cdot j\)

\[\theta(M\cdot n)\]

for \(y=1 \text{ to } M\cdot j\)

\[\theta(M\cdot n)\]

\[S_{j,x,y} = S_{j-1,x-1,x-A_j,y} \lor S_{j-1,x,y-A_j}\]

Check if any \(S_{n\cdot n/2,x,y}\) is true for \(x,y \leq M\cdot n\)
\[ S_{j,k,x,y} = S_{j-1,k-1,x-A_j,y} \lor S_{j-1,k,x,y-A_j} \]

**GERRYMANDER(P,A,m)**

initialize array S[0,0,0,0]
for j=1,...,n
  for k=1,...,n/2
    for x=0,...,jm
      for y=0,...,jm
        fill table according to equation
        search for true entry at S[n,n/2,>mn/4,>mn/4]
        \[ S_{j,k,x,y} = S_{j-1,k-1,x-A_j,y} \lor S_{j-1,k,x,y-A_j} \]
Scheduling

A new technique, Greedy
<table>
<thead>
<tr>
<th>Course Code</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>sy333</td>
<td>2</td>
<td>3.25</td>
</tr>
<tr>
<td>en162</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>ma123</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>cs4102</td>
<td>3.5</td>
<td>4.75</td>
</tr>
<tr>
<td>cs4402</td>
<td>4</td>
<td>5.25</td>
</tr>
<tr>
<td>cs6051</td>
<td>4.5</td>
<td>6</td>
</tr>
<tr>
<td>sy333</td>
<td>5</td>
<td>6.5</td>
</tr>
<tr>
<td>cs1011</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>
PROBLEM STATEMENT

\((a_1, \ldots, a_n)\)
\((s_1, s_2, \ldots, s_n)\)
\((f_1, f_2, \ldots, f_n)\)

(sorted) \(s_i < f_i\)

(Compatible)

FIND LARGEST SUBSET OF ACTIVITIES \(C=\{a_i\}\) SUCH THAT

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

\[ f_i < s_j \]
PROBLEM STATEMENT

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

\text{(COMPATIBLE)}

FIND LARGEST SUBSET OF ACTIVITIES \(C=\{a_i\}\) SUCH THAT

\(a_i, a_j \in C, i < j\)

\(f_i \leq s_j\)
PROBLEM STATEMENT

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

$Best_m$: most number of classes that can be scheduled between event $0$ and event $m$.

$Best_m = \max \left\{ 1 + Best_s(e_{2n}), Best_{e_{2n-1}} \right\}$
DYNAMIC PROGRAMMING

\[
\text{BEST } f_n = \max \ \text{BEST}_{s_n} + 1 \quad a_n \text{ IN:}
\]
\[
\text{BEST } e_t \quad a_n \text{ OUT:}
\]
GREEDY SOLUTION:

definition:

DEFINITION:

\[ \text{SOLTN}_{i,j} \rightarrow \text{maximal set of activities for period } c_i, j \]
GREEDY SOLUTION:

SOLTN_{i,j}

GOAL: SOLTN_{0,2n}
CLAIM: 

THE FIRST ACTION TO FINISH IN $e[i,j]$ IS ALWAYS PART OF SOME $SOLTN_{i,j}$

"first-to-finish is always part of the solution"
CLAIM: THE FIRST ACTION TO FINISH IN $e[i,j]$ IS ALWAYS PART OF SOME $SOLTN_{i,j}$

PROOF: Consider the optimal $SOLTN_{i,j}$. Let $a^*$ to be the $f$-to-$f$ in period $[i,j]$. Suppose that $a^* \notin SOLTN_{i,j}$.

Let $a$ be the first activity in $SOLTN_{i,j}$.

$\Rightarrow a^*$ is $f$-to-$f$, so $f(a^*) < f(a)$

$\Rightarrow SOLTN_{i,j} - \{a\} \cup \{a^*\}$ is also optimal.
GREEDY SOLUTION:

ALGORITHM:

1. Find first event to finish.
2. Add to solution.
3. Remove conflicting events.
4. 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.
**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.

**Handwritten note:** Much simpler algorithm. I pass thru the sorted list.
Greedy Solution:

Algorithm:
1. Find first event to finish.
2. Add to solution.
3. Remove conflicting events.
4. Continue.
RUNNING TIME

ALGORITHM: FIND FIRST EVENT TO FINISH. ADD TO SOLUTION.
REMOVE CONFLICTING EVENTS.
CONTINUE.
CACHING
Cache

main memory

CPU

load r2, addr a
store r4, addr b

CACHE HIT
QUESTION:

How to manage the cache??

① Assumption: Spse we know the entire access sequence ahead of time !!

② cache is fully associative
PROBLEM STATEMENT

input: \( K \cdot \text{cache size}, d_1, d_2, \ldots, d_n \) RAM access pattern

output: least \# of cache misses. Must satisfy the memory requests

cache is fully associative
PROBLEM STATEMENT

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

output:  
min # of cache misses

cache is 
fully associative, line size is 1
CONTRAST WITH REALITY
BELADY EVICT RULE

"if you must evict, evict the entry that is accessed farthest—in-the-future"
EXAMPLE

cache

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

cache

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

cache

a
b
c
cache

a
b
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

Cache

1. a b c d a
2. a b e d
3. a e d b a e c e a

not Belady rule
Thm: Belady rule is optimal.
Schedule for access pattern \( d_1, d_2, \ldots, d_n \):

Reduced schedule:

Reduced schedule: Schedule in which \( \text{"evict x for y"} \) occurs at operation \( i \) only if \( d_i = y \).

\[
\text{misses}(\text{schedule } S) \geq \text{misses}(\text{reduced}(S))
\]
REDUCED SCHEDULE

Def:
EXCHANGE LEMMA

- Spsc some reduced schedule $S$ agrees with $S_{ff}$ for the first $j$ operations.

Then $\exists$ a reduced schedule $S'$ that agrees with $S_{ff}$ on $j+1$ operations &

$$\text{misses}(S') \leq \text{misses}(S).$$
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 $S$. 
\( S^* \xrightarrow{\text{optimal schedule}} S_1 \xrightarrow{\text{lemma}} S_2 \xrightarrow{\text{lemma}} S_3 \cdots \xrightarrow{\text{lemma}} S_{\text{ff}} \)

\[
\text{misses}(S^*) = \text{misses}(S_1) \quad \Delta \quad \text{S_{ff} agrees w/ S_1 on 1 operation}
\]
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 # of misses as $S$. 
State of the cache after $J$ operations under the two schedules.

easy case 1

easy case 2
case 3

S
e
f

S_{eff}
TIMELINE

$S_f$

$S'$

$S$
Let access t
what if g=e?
what if \( g = f \)?
what if g is neither e nor f?
WHAT HAVE WE SHOWN

$S_{ff}$

$S'$

$S$
$S^*$

$S_{ff}$