4102

Sep 28 2009

abhi shelat

office hours: MONA. TUE 5-7
THU 5-7
FRI 8-10a
} CS LOUNGE.

abhi: WED 3-5 pm

midterm:

OCT 14th → OCT 16.

WED FRI.

① NO COLLABORATION.

NO INTERNET.

→

Reading

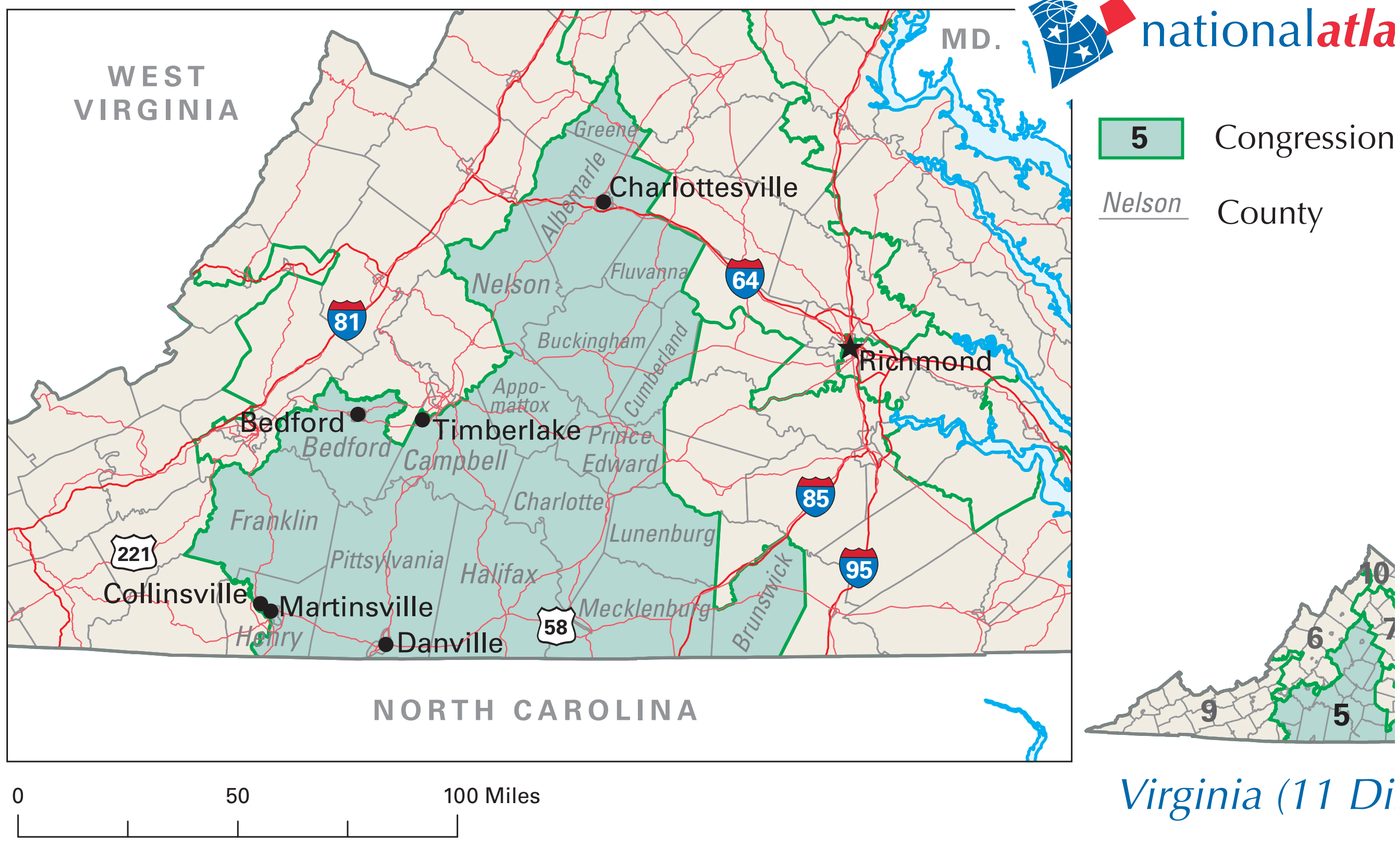
CLRS

15. DYNAMIC PROGRAMMING

16. GREEDY ALGORITHMS.

Gerrymander

Congressional District 5



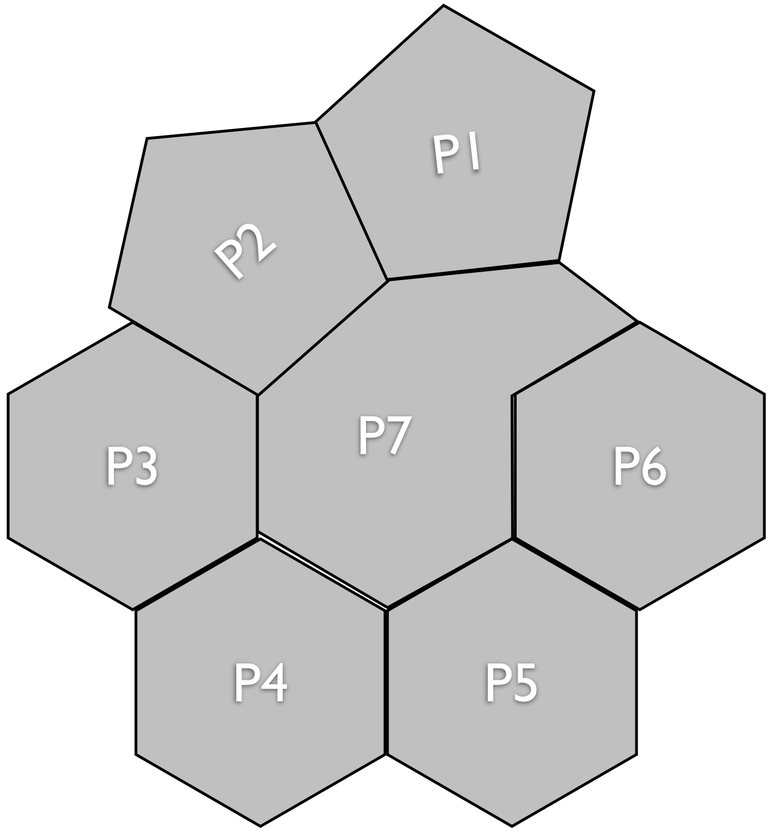
0 50 100 Miles

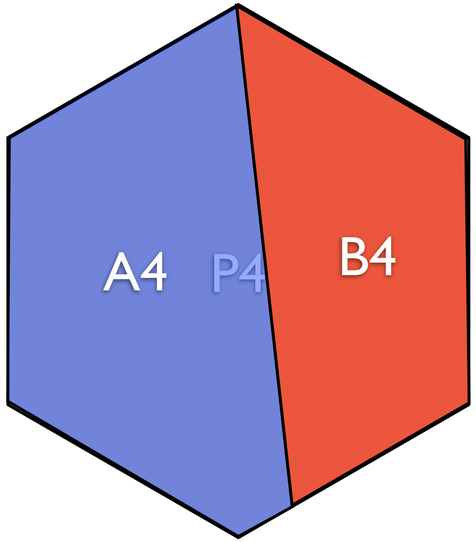
nationalatlas

5 Congression

Nelson County

Virginia (11 Di


P1
P2
P3
P7
P6
P4
P5


A4
P4
B4

GERRYMANDER PROBLEM

given: $P_1, P_2, \ldots, P_n$ - m people per precinct

$A_1, A_2, \ldots, A_n$ n is even

output: D_1, D_2

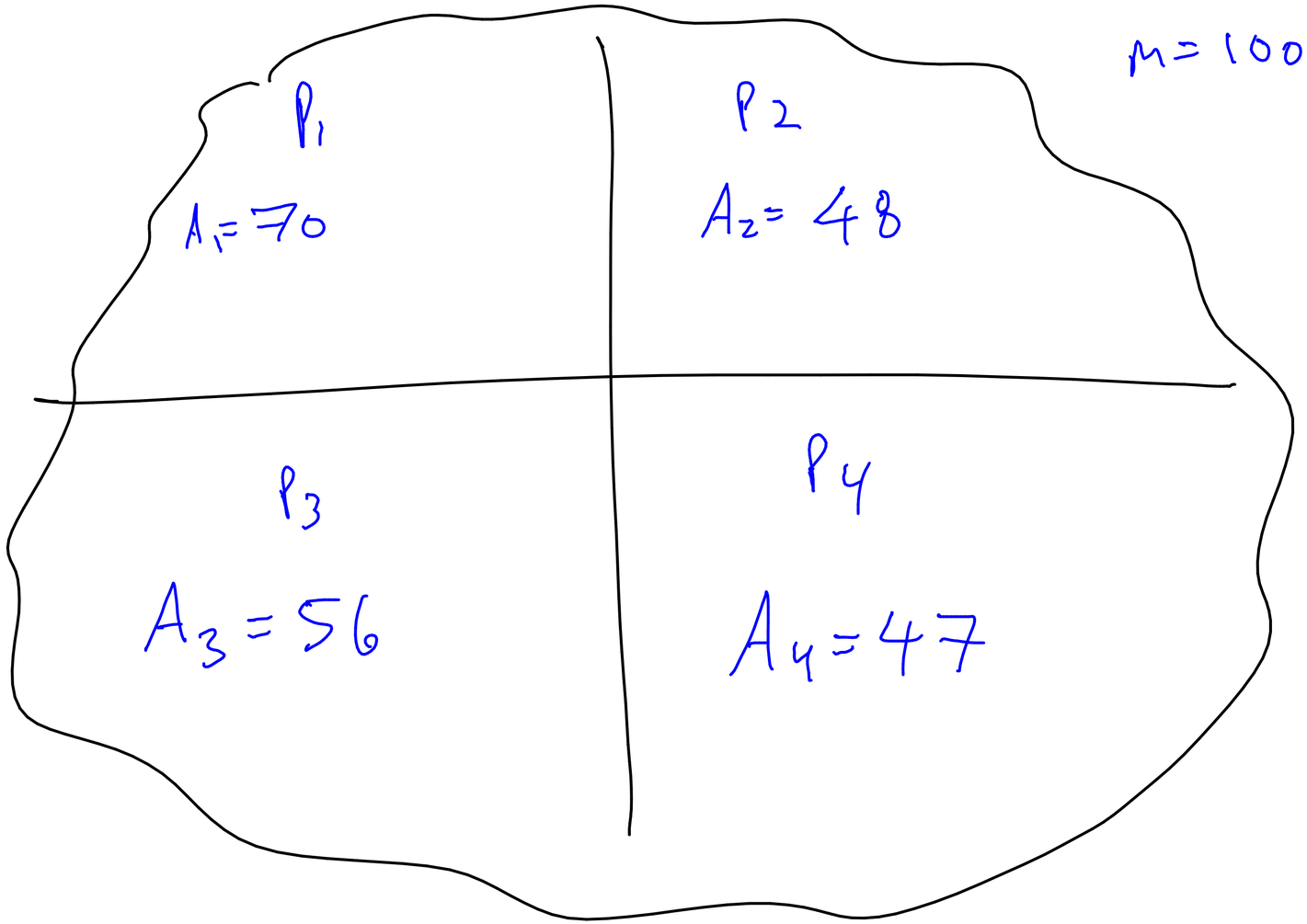
① $|D_1| = |P_2|$ → # precincts evenly split.

② $A(D_1) > \frac{m \cdot n}{4} = m\left(\frac{n}{2}\right) \cdot \frac{1}{2}$

③ $A(D_2) > \frac{m \cdot n}{4}$

if no solution, then algorithm should indicate failure.

EXAMPLE



$m = 100$

$D_1 = P_1 + P_2$ A

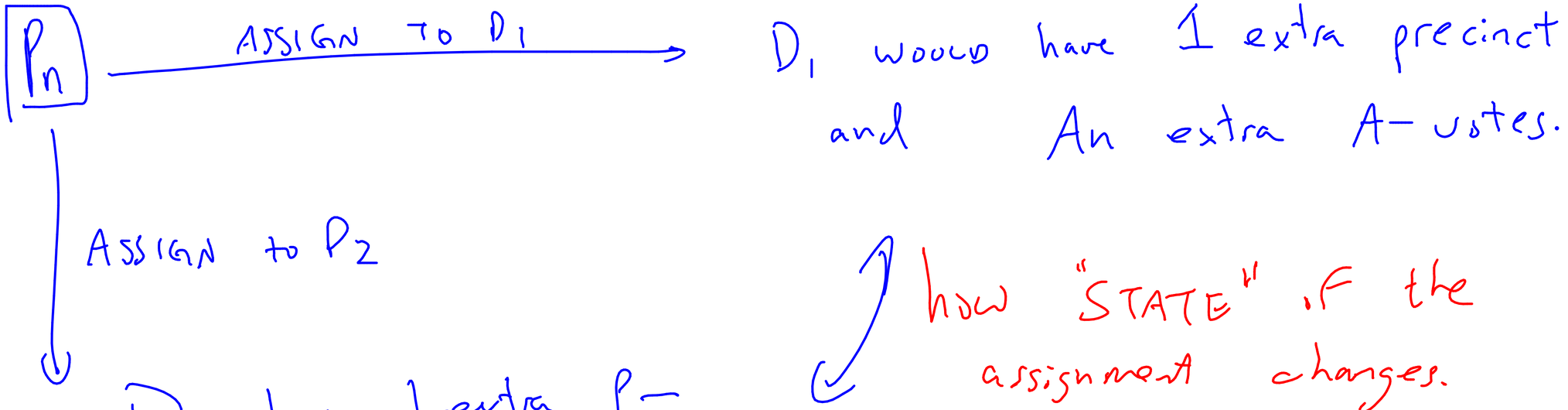
$D_2 = P_3 + P_4$ A

$D_1 = P_1 + P_3$ A

$D_2 = P_2 + P_4$ B.

GERRYMANDER

imagine very last precinct and how it is assigned:



GERRYMANDER

$S_{j,k,x,y} =$ TRUE IF there is a split of the first j precincts such that k of them are in D_1 and $A(D_1) = x$ $A(D_2) = y$.

GERRYMANDER

$S_{j,k,x,y}$ = there is a split of first **j** precincts
in which |D1|=**k** and
x people in D1 vote A
y people in D2 vote A

$S_{j,k,x,y} = S_{j-1,k-1,x-A_j,y}$ OR $S_{j-1,k,x,y-A_j}$

$S_{j-1,k-1,x-A_j,y}$: & we assign A_j to D_1, then

$S_{j-1,k,x,y-A_j}$: assign A_j to D_2

this will be true.

$$S_{j,k,x,y} = S_{j-1,k-1,x-A_j,y} \vee S_{j-1,k,x,y-A_j}$$

OR

GERRYMANDER(P,A,m)

initialize array S[o,o,o,o] = true.

for j=1 to n.

for k=0 to $\frac{n}{2}$ & k≤j

for x=0 to j·m

for y=0 to j·m

Set $S_{j,k,x,y}$ using formula.

$n \cdot n \cdot (n \cdot m) \cdot n \cdot m$

$\Theta(n^4 \cdot m^2)$

→ Search for entry $S[n, \frac{n}{2}, \frac{mn}{4}, \frac{mn}{4}]$ THAT IS TRUE. $O(n^2 \cdot m^2)$

$\frac{m \cdot n}{4}$ $\frac{m \cdot n}{4}$

Alternative Algorithm:

Brute force.

Try all $\binom{n}{n/2} > 2^{n/2}$ ways to split n precincts into two districts of eq size.

$$S_{j,k,x,y} = S_{j-1,k-1,x-A_j,y} \text{ OR } S_{j-1,k,x,y-A_j}$$

GERRYMANDER(P,A,m)

initialize array S[0,0,0,0]
for j=1,...,n
    k=0,...,n/2 and k<=j
    x=0,...,jm
    y=0,...,jm
        fill table according to equation
search for true entry at S[n,n/2, >mn/4, >mn/4]

GERRYMANDER(P,A,m)

initialize array S[0,0,0,0]
for j=1,...,n
    k=0,...,n/2 and k<=j
    x=0,...,jm
    y=0,...,jm
        fill table $S_{j,k,x,y} = S_{j-1,k-1,x-A_j,y}$ or $S_{j-1,k,x,y-A_j}$

search for true entry at S[n,n/2, >mn/4, >mn/4]

EXAMPLE

A_1 (6) A_2 (3) A_3 (5) A_4 (2) ..... $m = 10$.

S_{0000} TRUE

$S_{1000} = F$
$S_{1001} = F$
S_{1002} :
S_{1003}
:
$S_{1006} = T$
....
S_{10010}

S_{1100}
S_{1110}
S_{1160} = S_{0000} OR ——— = TRUE

S_{2000}
S_{2001}
:
$S_{2009} = T$
:

S_{2163} = TRUE.

AND SO ON ....

RECAP OF DYN PROG

① consider last step

② Define a SUBPROBLEM $Best_i$ $Cost_{(i,j)}$ OPT_k S_{jk+y}

SUBPROBLEM Equation.

③ order of evaluation

---

log, matrix chain, typesetting, scan, gerry.

EDIT DISTANCE BINARY TREE*


greedy

Scheduling

	START	END
sy333	2	3.25
en162	1	4
ma123	3	4
cs4102	3.5	4.75
cs4402	4	5.25
cs6051	4.5	6
sy333	5	6.5
cs1011	7	8

PROBLEM STATEMENT

$(a_1, \ldots, a_n)$ → list of activities

$(s_1, s_2, \ldots, s_n)$ → start times for activities

$(f_1, f_2, \ldots, f_n)$ (SORTED) $s_i < f_i$

↳ finish times

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

for any $a_i, a_j \in C$ $i < j$ $f_i < s_j$

"compatible activity set"

# of activities.

PROBLEM STATEMENT

$(a_1, \ldots, a_n)$

$(s_1, s_2, \ldots, s_n)$

$(f_1, f_2, \ldots, f_n)$ (SORTED) $\quad s_i < f_i$

(COMPATIBLE)

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

$$a_i, a_j \in C, i < j$$

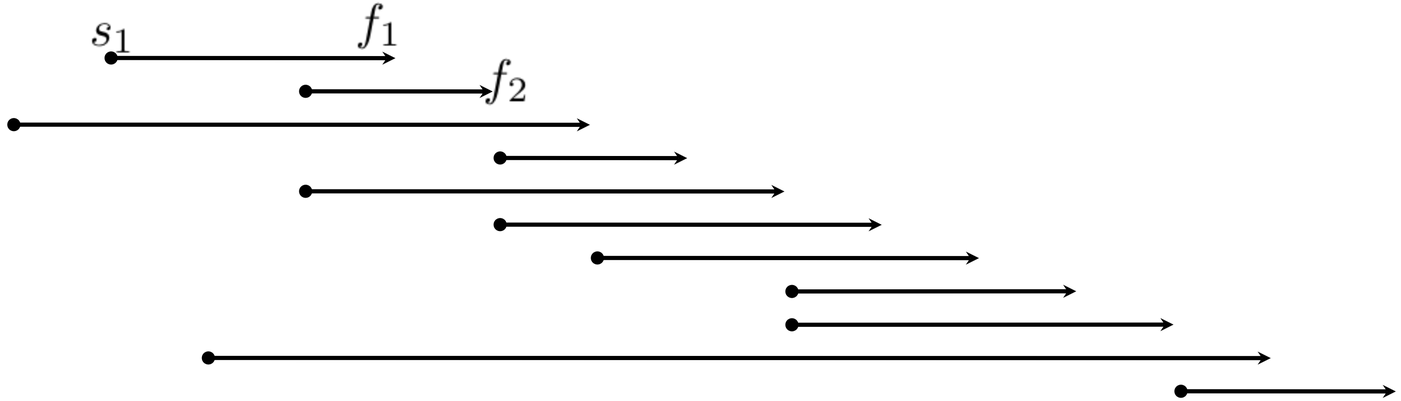
$$f_i \leq s_j$$

PROBLEM STATEMENT

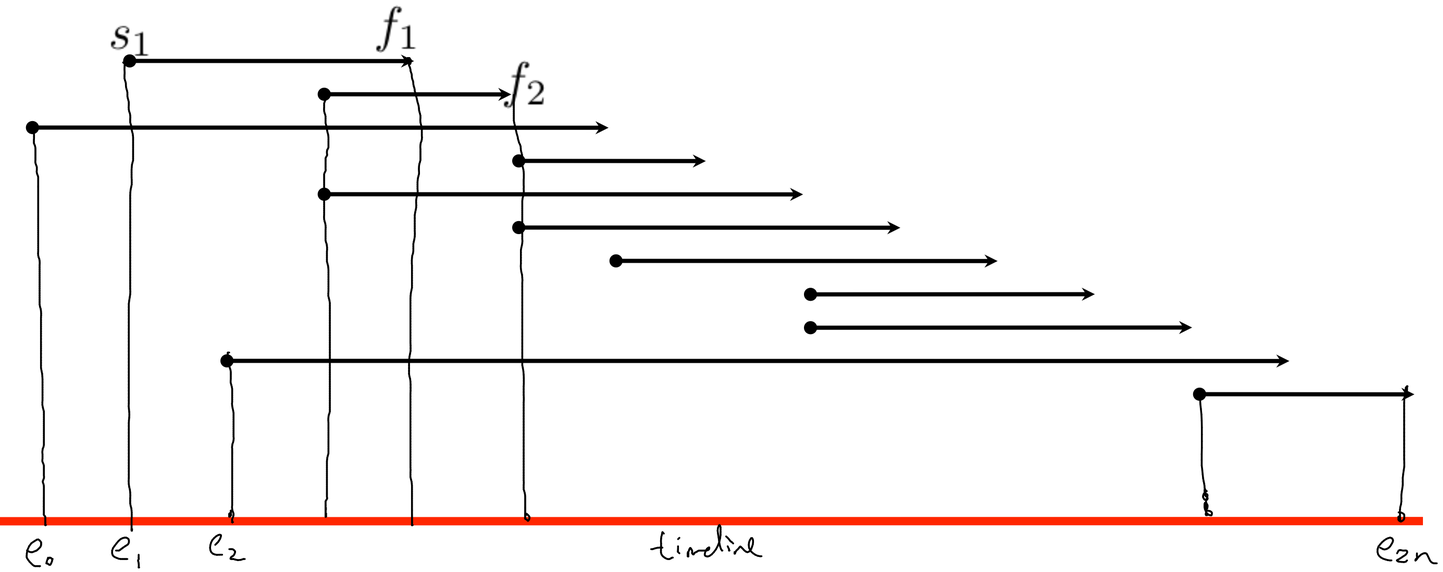
$(a_1, \ldots, a_n)$

$(s_1, s_2, \ldots, s_n)$

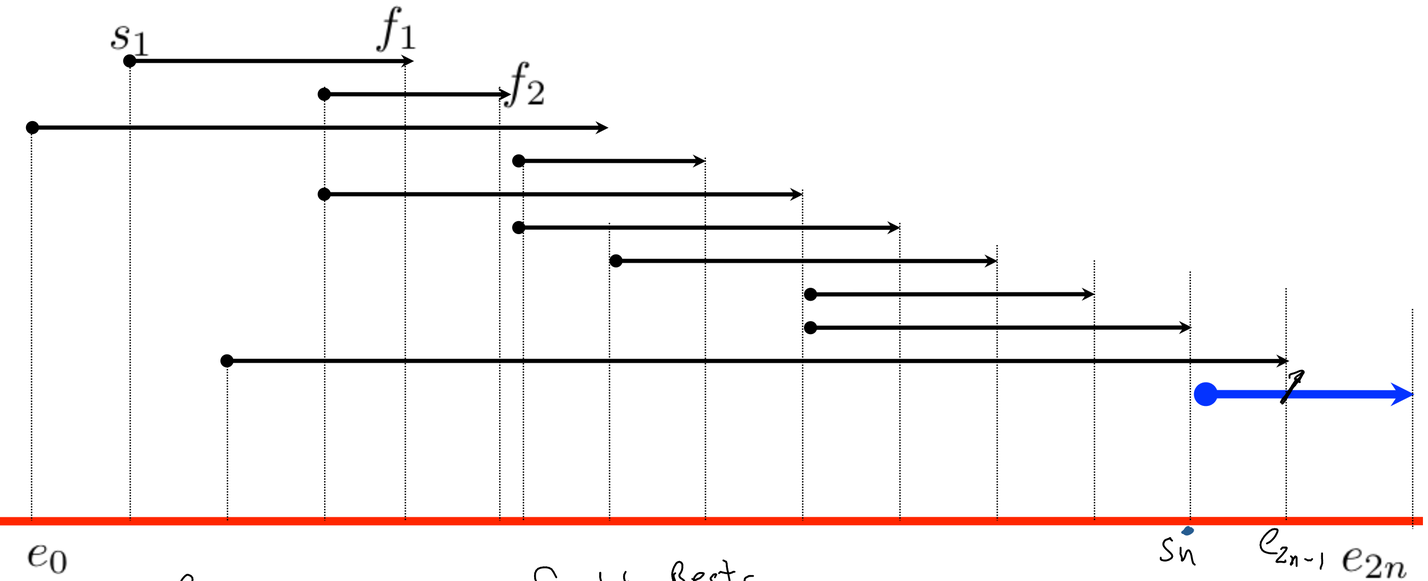
$(f_1, f_2, \ldots, f_n)$ (SORTED) $\quad s_i < f_i$



DYNAMIC PROGRAMMING

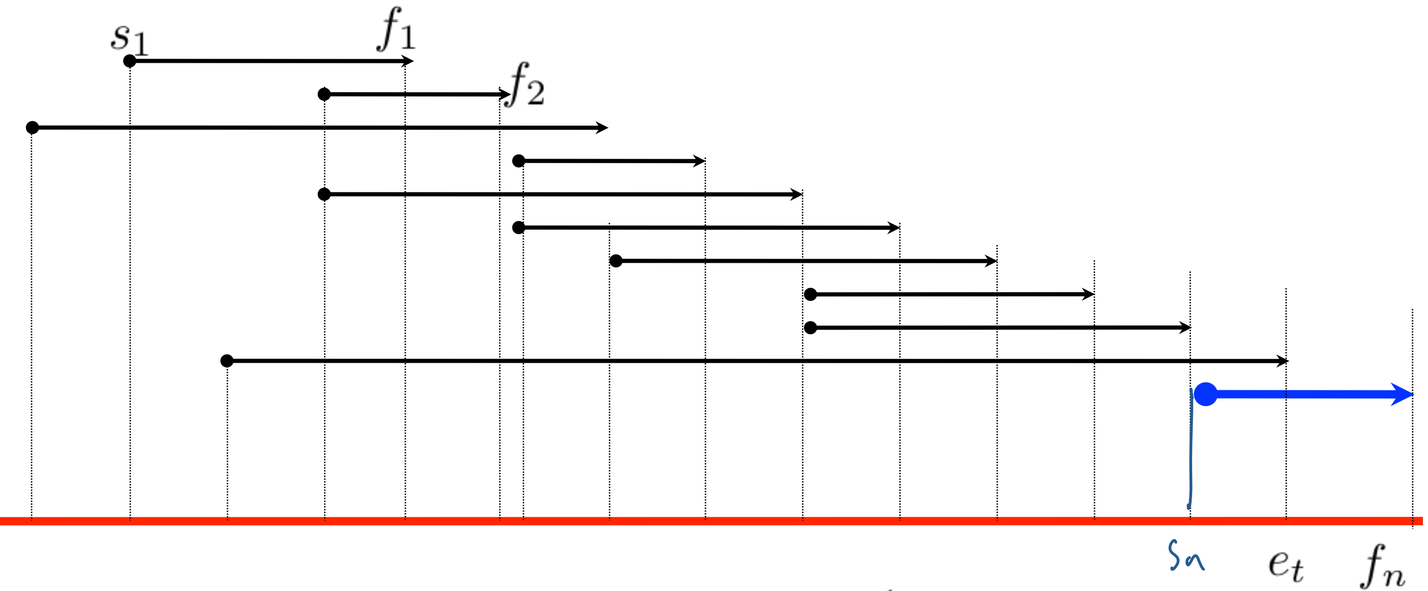


DYNAMIC PROGRAMMING



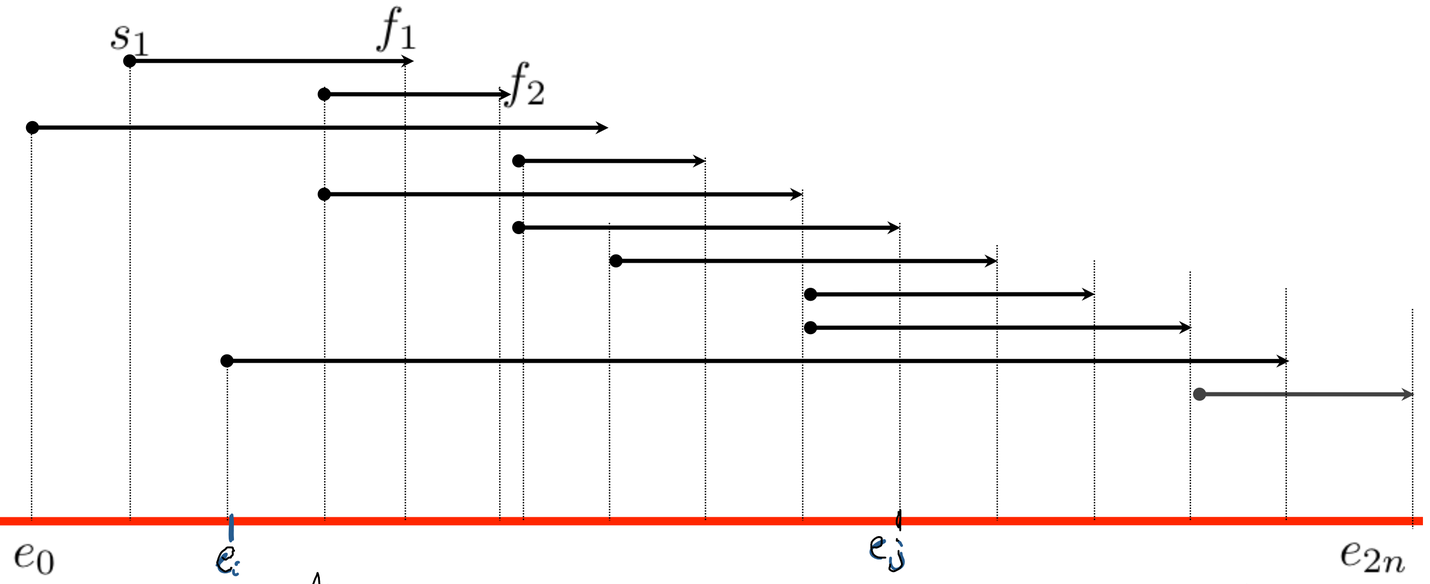
$$Best_{e_{2n}} = \max \begin{cases} \text{If } Best_{s_n} \\ Best_{e_{2n-1}} \end{cases}$$

DYNAMIC PROGRAMMING



$$\text{BEST}_{f_n} = \begin{matrix}\text{MAX} \\ \\ \end{matrix} \begin{matrix}\text{BEST}_{s_n} + 1 & a_n \text{ IN:} \\ \text{BEST}_{e_t} & a_n \text{ OUT:}\end{matrix}$$

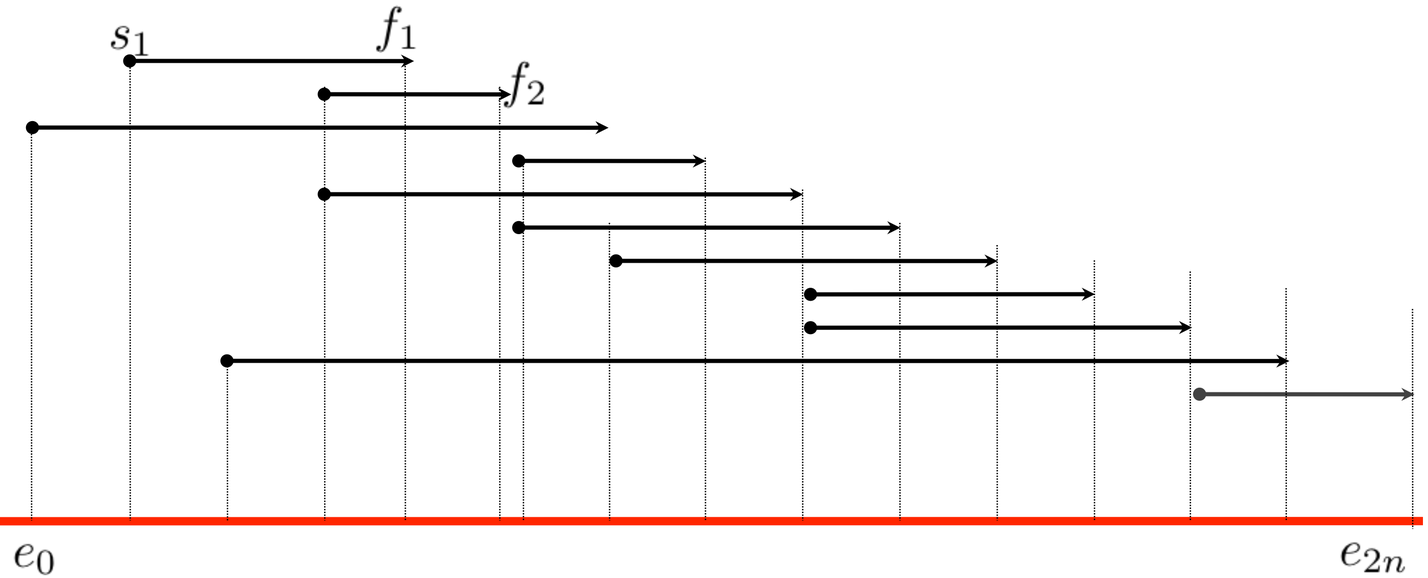
GREEDY SOLUTION:



DEFINITION: $\text{SOLTN}_{i,j}$ A MAXIMAL SET OF COMPATIBLE ACTIVITIES that occur in $[e_i, e_j]$

GOAL: $\text{SOLTN}_{0,2n}$ (find some $\text{SOLTN}_{0,2n}$)

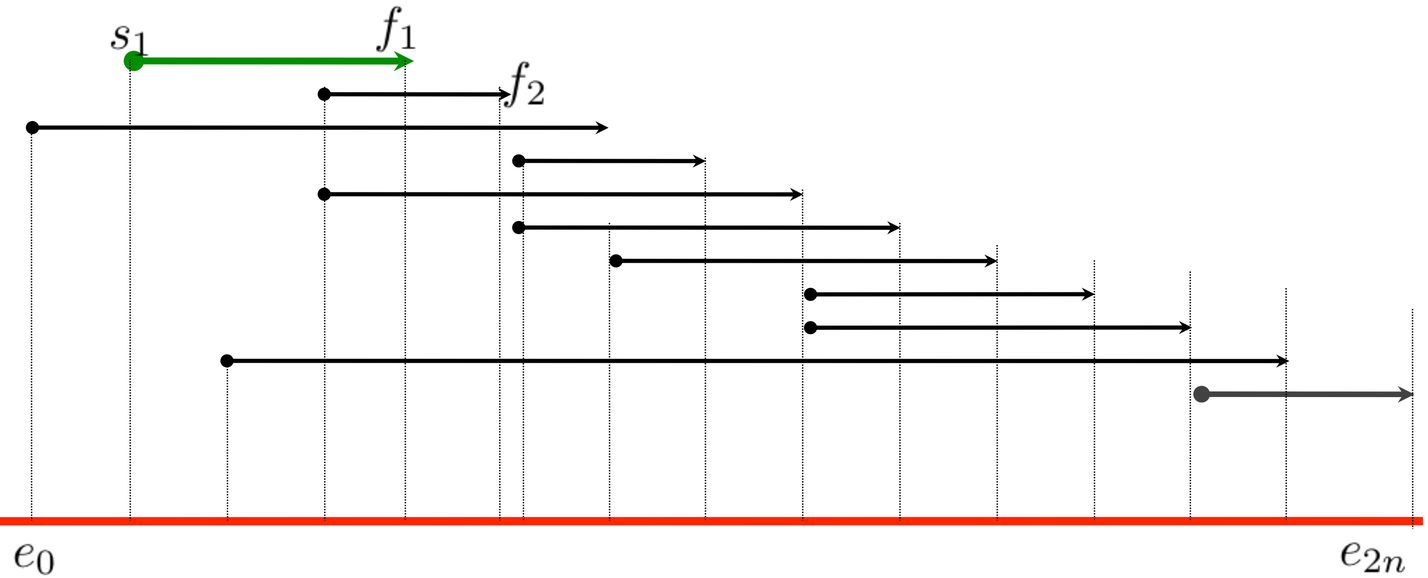
GREEDY SOLUTION:



$\text{SOLTN}_{i,j}$

GOAL: $\text{SOLTN}_{0,2n}$

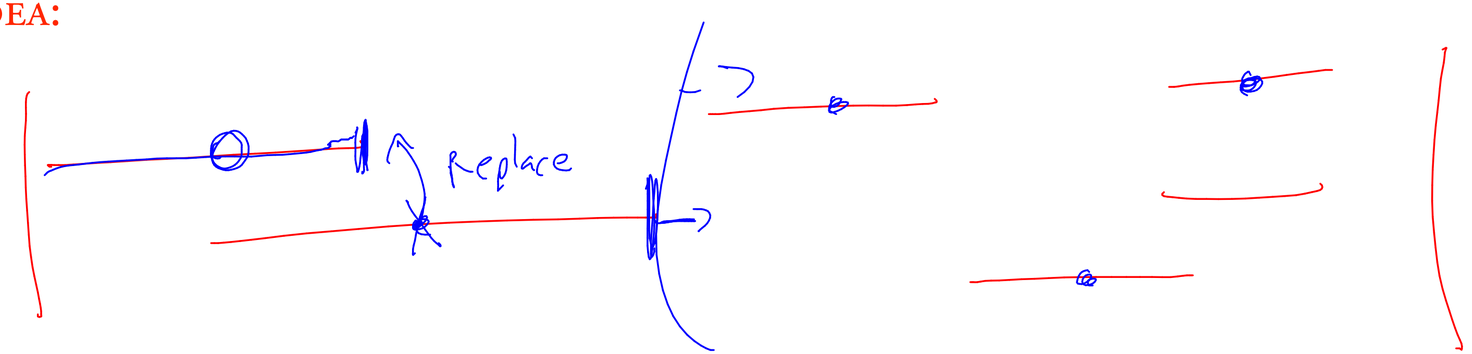
GREEDY SOLUTION:



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 $\text{SOLTN}_{i,j}$

IDEA:



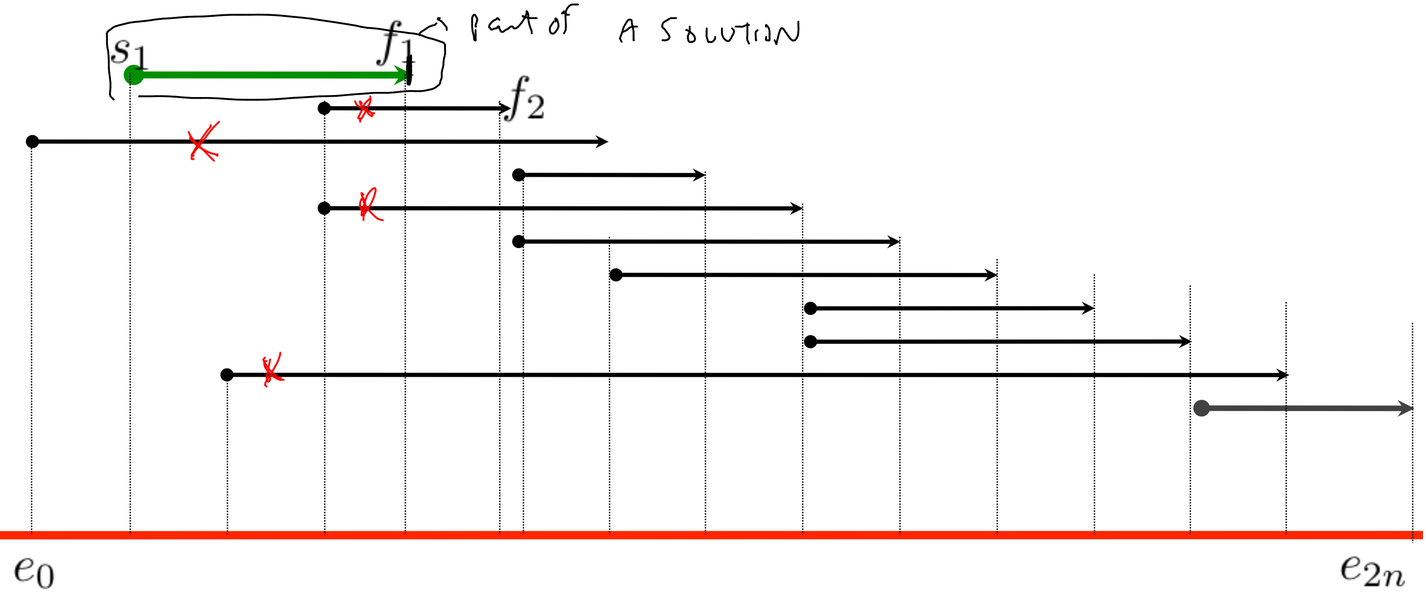
TIME Period

- optimal Solution.

CLAIM: THE FIRST ACTION TO FINISH IN `e[i,j]` IS ALWAYS PART OF SOME $\mathrm{SOLTN}_{i,j}$

PROOF:

GREEDY SOLUTION:

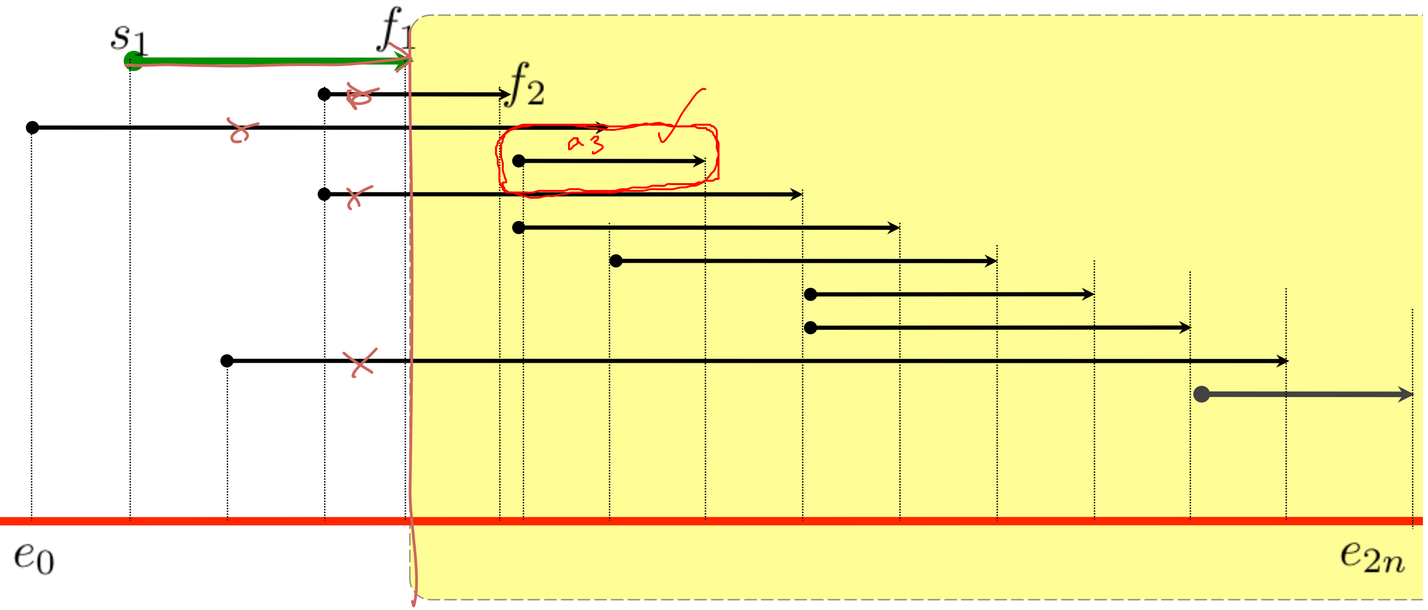


ALGORITHM: FIND FIRST EVENT TO FINISH. ADD TO SOLUTION.

REMOVE CONFLICTING EVENTS.

CONTINUE.

GREEDY SOLUTION:

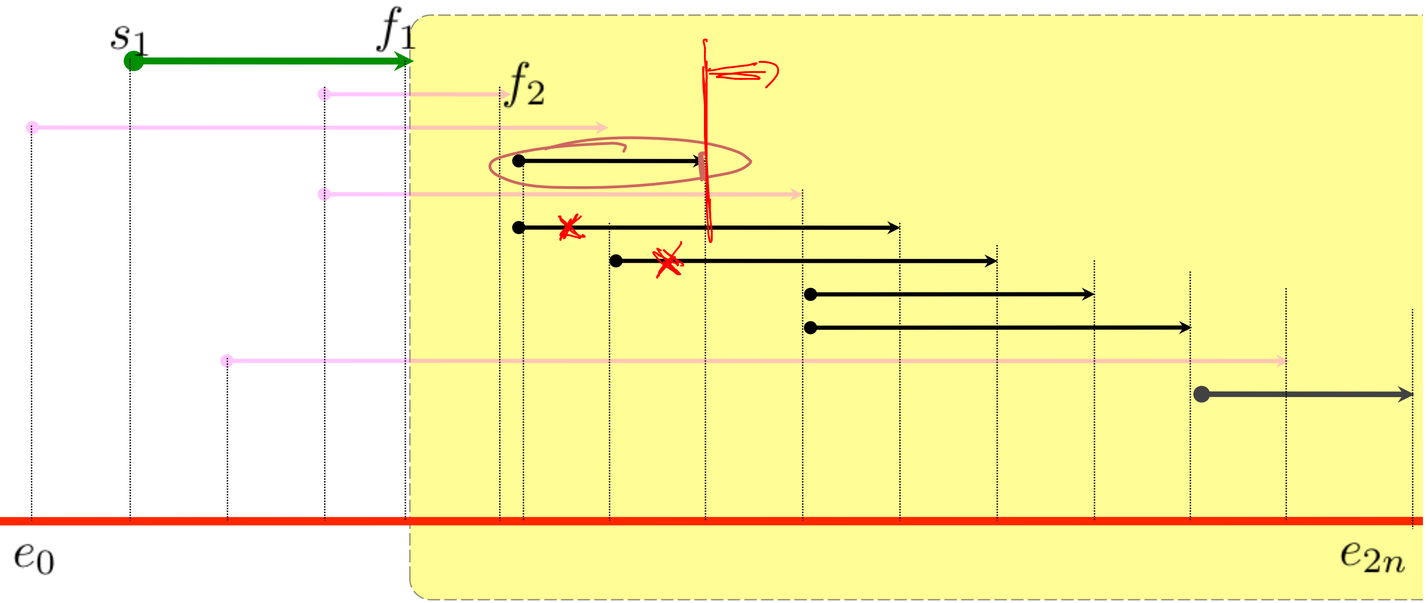


ALGORITHM: FIND FIRST EVENT TO FINISH. ADD TO SOLUTION.

REMOVE CONFLICTING EVENTS.

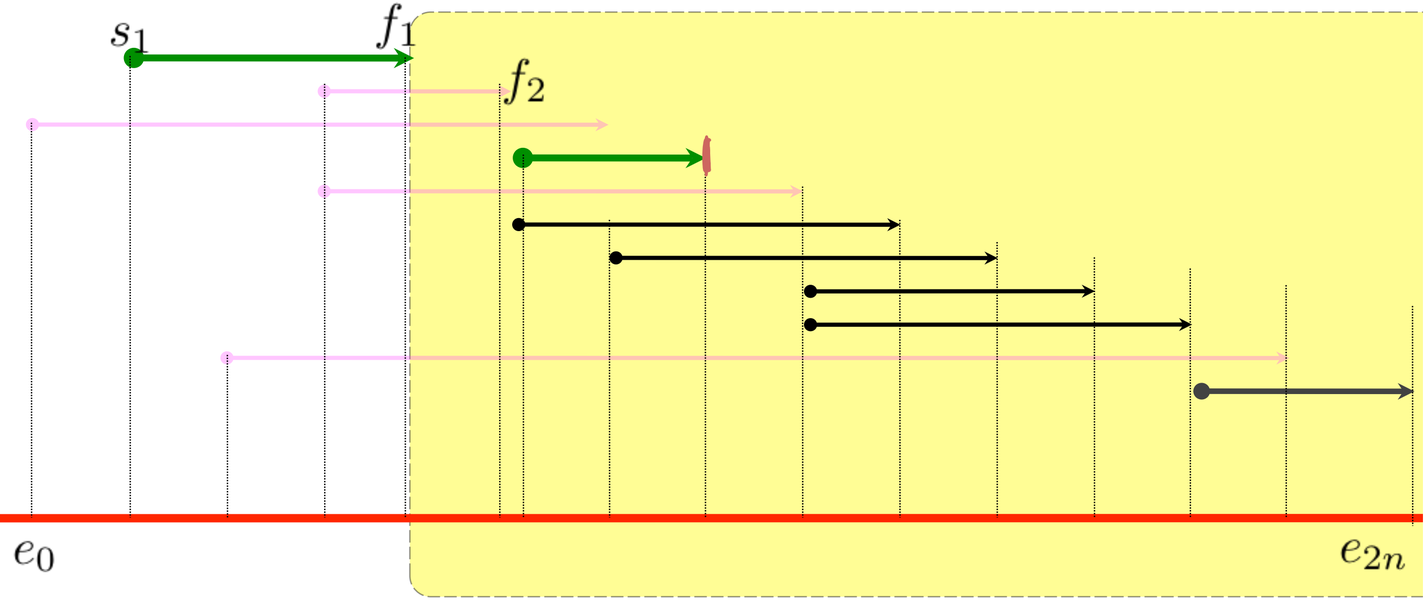
CONTINUE.

GREEDY SOLUTION:



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

GREEDY SOLUTION:

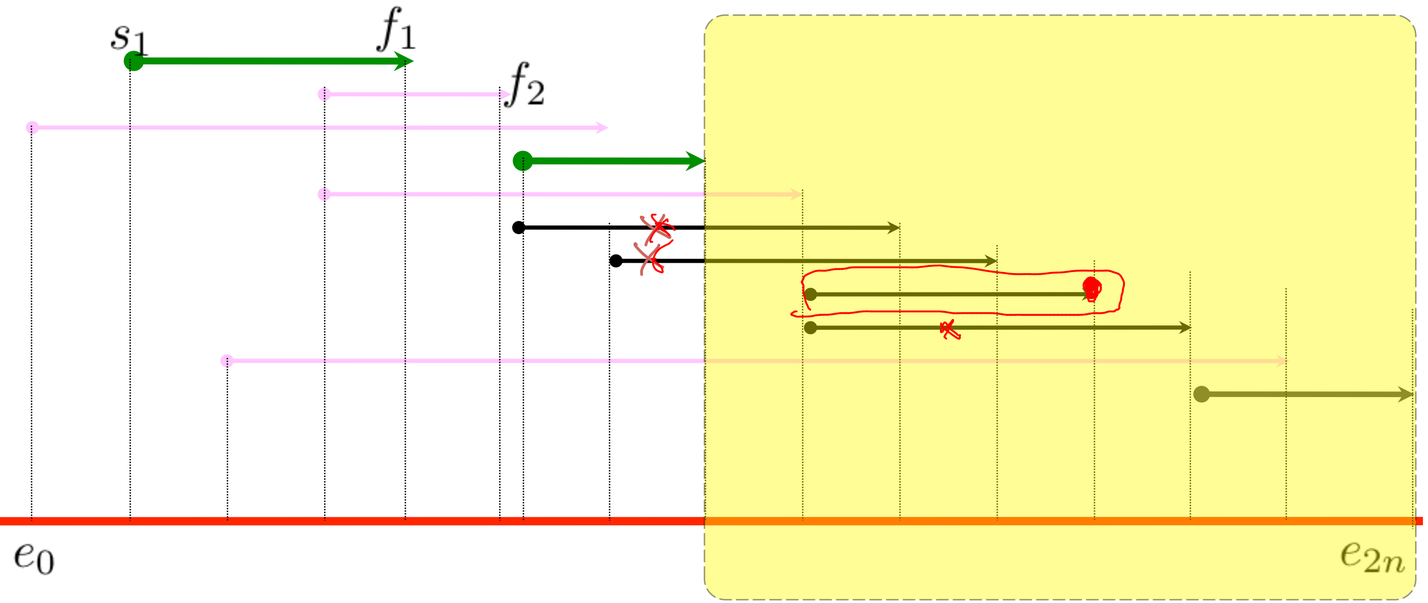


ALGORITHM: FIND FIRST EVENT TO FINISH. ADD TO SOLUTION.

REMOVE CONFLICTING EVENTS.

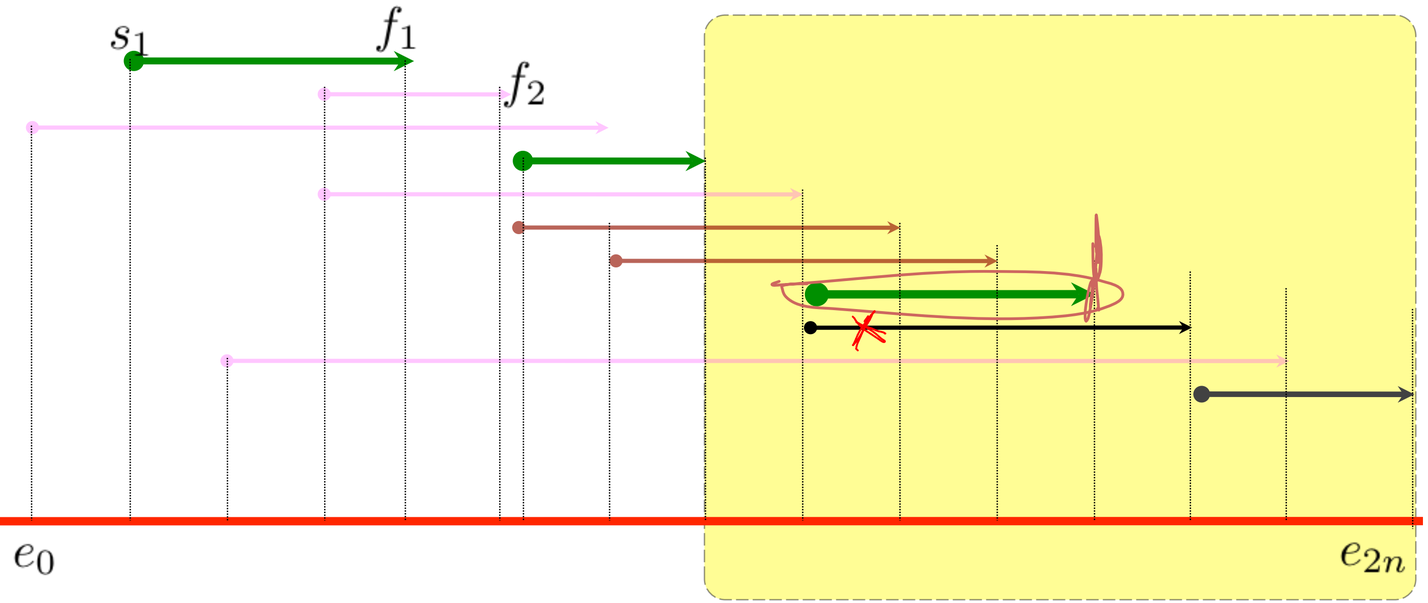
CONTINUE.

GREEDY SOLUTION:



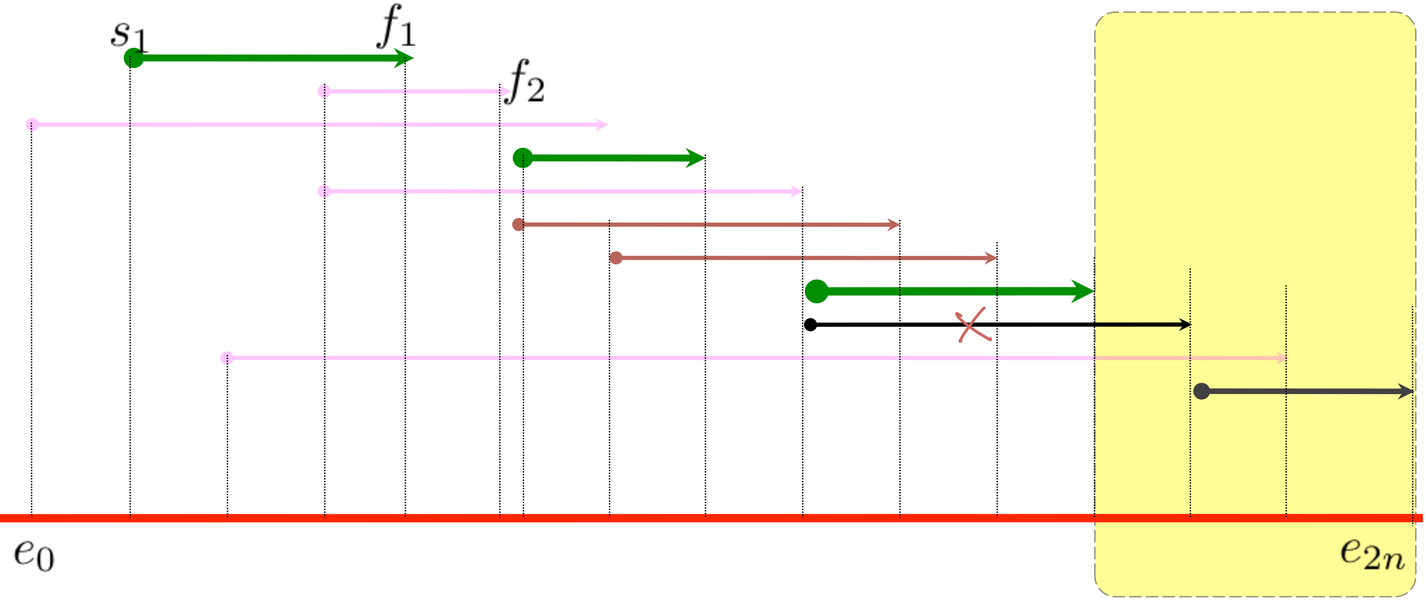
ALGORITHM: FIND FIRST EVENT TO FINISH. ADD TO SOLUTION.
REMOVE CONFLICTING EVENTS.
CONTINUE.

GREEDY SOLUTION:



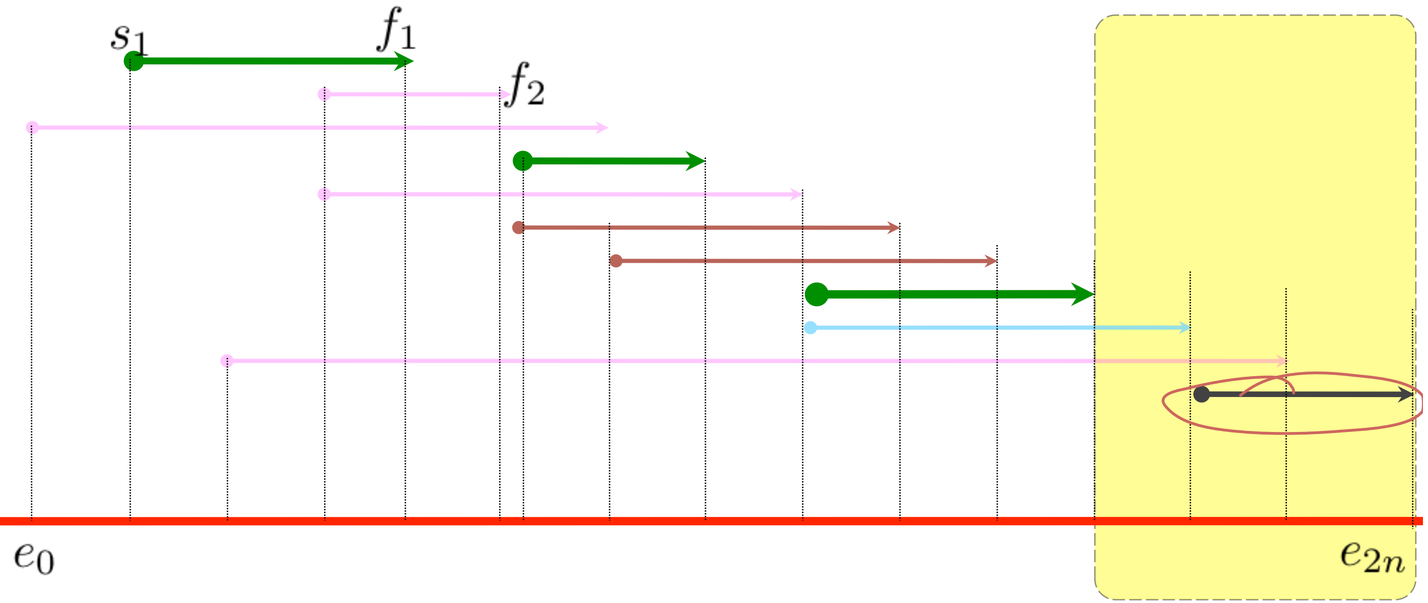
ALGORITHM: FIND FIRST EVENT TO FINISH. ADD TO SOLUTION.
REMOVE CONFLICTING EVENTS.
CONTINUE.

GREEDY SOLUTION:



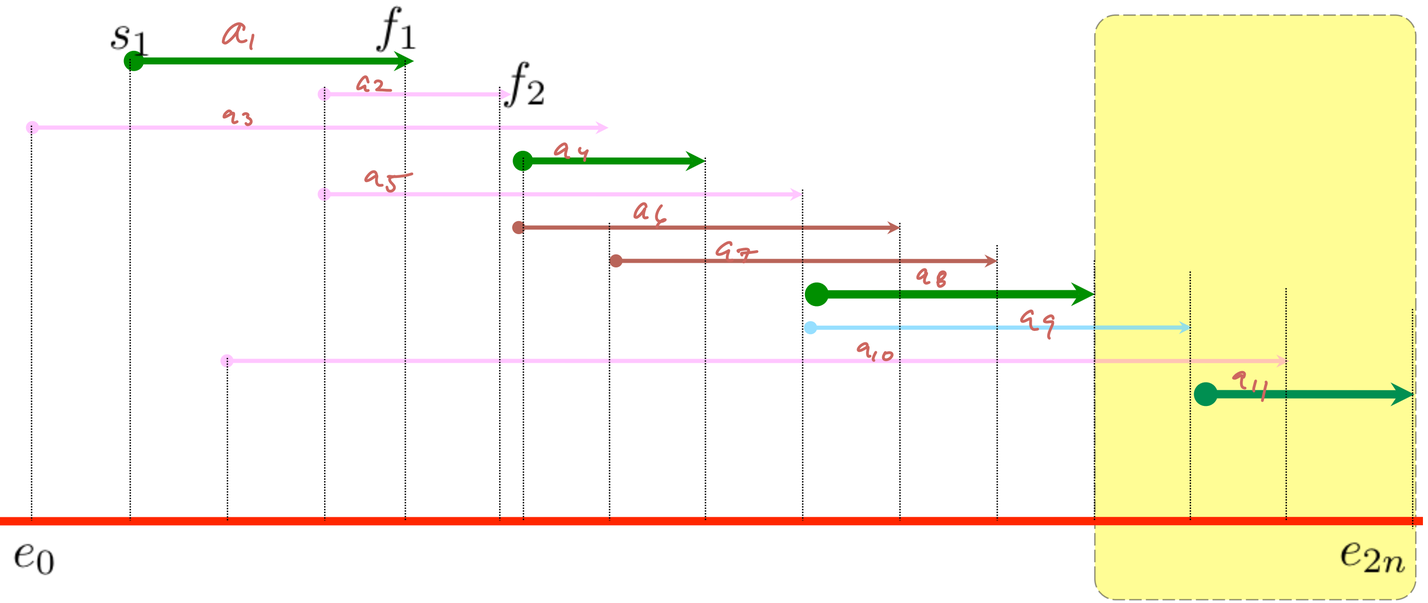
ALGORITHM: FIND FIRST EVENT TO FINISH. ADD TO SOLUTION.
REMOVE CONFLICTING EVENTS.
CONTINUE.

GREEDY SOLUTION:



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

GREEDY SOLUTION:

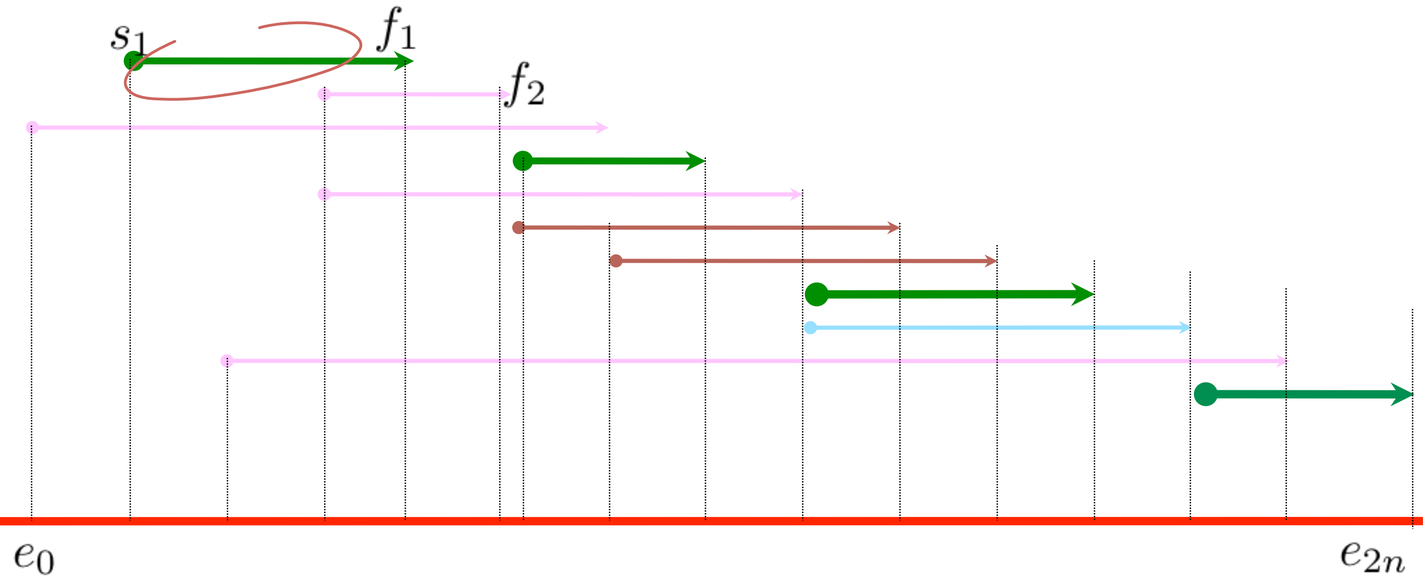


ALGORITHM: FIND FIRST EVENT TO FINISH. ADD TO SOLUTION.

REMOVE CONFLICTING EVENTS.

CONTINUE.

GREEDY SOLUTION:



ALGORITHM: FIND FIRST EVENT TO FINISH. ADD TO SOLUTION.

REMOVE CONFLICTING EVENTS.

CONTINUE.

RUNNING TIME

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