# W11: Greedy Algorithms

CS3230 AY21/22 Sem 2

# Table of Contents

Click on the link to jump to the relevant sections!

# Greedy Algorithms

# Recall: How to DP?

Brute Force, but *carefully*

1. Identify the subproblems
2. To solve the current subproblem, **assume** you have solved the other (smaller) subproblems
3. Relate the smaller subproblem to the current subproblem
   a. Guess the relation!
   b. This might involve trying **all** subproblems!

- Your subproblem result might be re-used -- store it in a table!
- Time complexity: total time to compute **all subproblems**

# How to greedy?

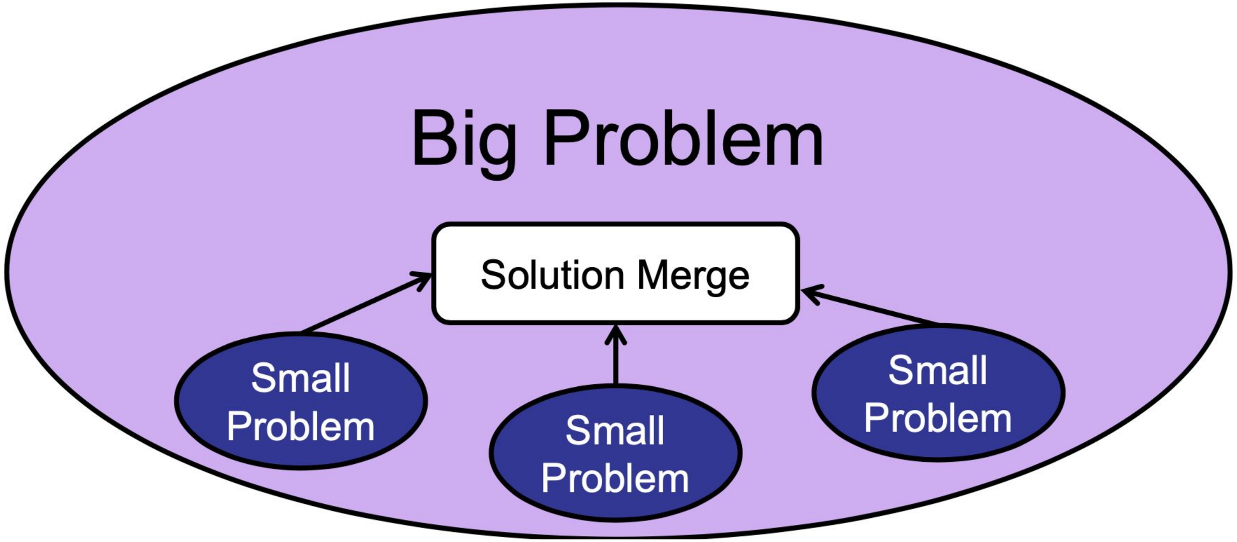
Brute Force, but *even more carefully*

1. Identify the subproblems
2. To solve the current subproblem, **assume** you have solved the other (smaller) subproblems
3. Relate the smaller subproblem to the current subproblem
    a. Guess the relation!
    b. ~~This might involve trying **all** subproblems!~~ Pick the “best” subproblem

- ~~Your subproblem result might be re-used -- store it in a table!~~
- Time complexity: total time to compute **all subproblems** (Usually, one subproblem takes O(1)) time. But there might be pre-processing

# When to use DP and Greedy?

**Optimal substructure**: Optimal solutions can be reconstructed from smaller subproblems



# Important thing to do in Greedy Algorithm

- In DP: try **all** the subproblems
- In Greedy: try **one** subproblem, chosen **greedily** (usually, something like the one that gives the max/min value)

- You have to prove that the greedy choice will still give you the best optimal solution
  - Usually by the ‘cut-and-paste’ argument
  - ‘Cut’ out the current optimal solution and ‘paste’ a solution using greedy choice
    - Show that the solution stays as “good” (doesn’t become “worse”)

# Question 1: Optimal Substructure of pairing files

# Question 1

- Bob has music files that he wants to burn into CDs

# Question 1

- Bob has music files that he wants to burn into CDs
- CD storage capacity = 100 MB

# Question 1

- Bob has music files that he wants to burn into CDs
- CD storage capacity = 100 MB
- Cannot split music file, i.e., no burning of single file to more than 1 CD
- Not more than two music files per CD

# Question 1

- Bob has music files that he wants to burn into CDs
- CD storage capacity = 100 MB
- Cannot split music file, i.e., no burning of single file to more than 1 CD
- Not more than two music files per CD
- Given set A of file sizes, each smaller than 100MB, let `MinCD(A)` denote the minimum number of CDs required to fit the files described in A

# Question 1

- Bob has music files that he wants to burn into CDs
- CD storage capacity = 100 MB
- Cannot split music file, i.e., no burning of single file to more than 1 CD
- Not more than two music files per CD
- Given set A of file sizes, each smaller than 100MB, let `MinCD(A)` denote the minimum number of CDs required to fit the files described in A

Which describes optimal substructure?

**Assuming at least one pair of files fit**

# Question 1

- Bob has music files that he wants to burn into CDs
- CD storage capacity = 100 MB
- Cannot split music file, i.e., no burning of single file to more than 1 CD
- Not more than two music files per CD
- Given set A of file sizes, each smaller than 100MB, let `MinCD(A)` denote the minimum number of CDs required to fit the files described in A

Set A, without $f_1$ and $f_2$

Which describes optimal substructure?
**Assuming at least one pair of files fit**

1. For any pair of files $f_1, f_2$ in $A$
   $MinCD(A) = 1 + MinCD(\boxed{A \setminus \{f_1, f_2\}})$

2. For any pair of files $f_1, f_2$ in $A$ that belong on a single CD in an optimal solution,
   $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$

3. If $f_1$ and $f_2$ are the largest and smallest files in $A$,
   $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$

# Question 1 Solution

1. For any pair of files $f_1$ and $f_2$ in A, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$
2. For any pair of files $f_1$ and $f_2$ in A that belong on a single CD in an optimal solution, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$
3. If $f_1$ and $f_2$ are the largest and smallest in A, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$

Option 1 **does not work** - we cannot simply pair **ANY** two file and still be optimal

Counterexample: `A = {10, 20, 80, 90}`

Optimal Solution: `MinCD(A) = 2` [because group `{10, 90}, {20, 80}`]

# Question 1 Solution

1. For any pair of files $f_1$ and $f_2$ in A, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$
2. For any pair of files $f_1$ and $f_2$ in A that belong on a single CD in an optimal solution, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$
3. If $f_1$ and $f_2$ are the largest and smallest in A, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$

Option 1 **does not work** - we cannot simply pair **ANY** two file and still be optimal

Counterexample: `A = {10, 20, 80, 90}`

Optimal Solution: `MinCD(A) = 2` [because group `{10, 90}, {20, 80}`]

BUT choose **ANY** pair, e.g. {10, 20}

`MinCD({10, 20, 80, 90}) ≠ 1 + MinCD({80, 90})`

# Question 1 Solution

1. For any pair of files $f_1$ and $f_2$ in A, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$
2. For any pair of files $f_1$ and $f_2$ in A that belong on a single CD in an optimal solution, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$
3. If $f_1$ and $f_2$ are the largest and smallest in A, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$

Option 1 **does not work** - we cannot simply pair **ANY** two file and still be optimal

Counterexample: `A = {10, 20, 80, 90}`

Optimal Solution: `MinCD(A) = 2` [because group `{10, 90}, {20, 80}`]

BUT choose **ANY** pair, e.g. {10, 20}

`MinCD({10, 20, 80, 90}) ≠ 1 + MinCD({80, 90})`

`MinCD({80, 90}) = 2` [because `{80}` and `{90}` are alone]

# Question 1 Solution

1. For any pair of files $f_1$ and $f_2$ in A, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$
2. For any pair of files $f_1$ and $f_2$ in A that belong on a single CD in an optimal solution, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$
3. If $f_1$ and $f_2$ are the largest and smallest in A, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$

Option 1 **does not work** - we cannot simply pair **ANY** two file and still be optimal

Counterexample: `A = {10, 20, 80, 90}`

Optimal Solution: `MinCD(A) = 2` [because group `{10, 90}, {20, 80}`]

BUT choose **ANY** pair, e.g. {10, 20}

`MinCD({10, 20, 80, 90}) ≠ 1 + MinCD({80, 90})`
2 ≠ 1 + 2

`MinCD({80, 90}) = 2` [because `{80}` and `{90}` are alone]

# Question 1 Solution

1. For any pair of files $f_1$ and $f_2$ in A, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$
2. For any pair of files $f_1$ and $f_2$ in A that belong on a single CD in an optimal solution, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$
3. If $f_1$ and $f_2$ are the largest and smallest in A, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$

Option 3 **does not work** - largest and smallest file may not fit into a single CD

Counterexample: A `= {10, 20, 30, 95, 99}`

Optimal: MinCD(A) = 4 [`{10, 20}, {30}, {95}, {99}` is a possible optimal]

# Question 1 Solution

1. For any pair of files $f_1$ and $f_2$ in A, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$
2. For any pair of files $f_1$ and $f_2$ in A that belong on a single CD in an optimal solution, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$
3. If $f_1$ and $f_2$ are the largest and smallest in A, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$

Option 3 **does not work** - largest and smallest file may not fit into a single CD

Counterexample: A `= {10, 20, 30, 95, 99}`

Optimal: MinCD(A) = 4 [`{10, 20}, {30}, {95}, {99}` is a possible optimal]

BUT choose largest and smallest pair, e.g. `{10, 99}`

This doesn't even fit!

```
MinCD(A) ≠ 1 + MinCD({20, 30, 95})
4 ≠ 1 + 2
```

# Question 1 Solution

1. For any pair of files $f_1$ and $f_2$ in A, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$
2. For any pair of files $f_1$ and $f_2$ in A that belong on a single CD in an optimal solution, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$
3. If $f_1$ and $f_2$ are the largest and smallest in A, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$

Option 3 **does not work** - largest and smallest file may not fit into a single CD

Counterexample: A `= {10, 20, 30, 95, 99}`

Optimal: MinCD(A) = 4 [`{10, 20}, {30}, {95}, {99}` is a possible optimal]

BUT choose largest and smallest pair, e.g. `{10, 99}`

This doesn't even fit!

`MinCD(A) ≠ 1 + MinCD({20, 30, 95})`
`4 ≠ 1 + 2`

`MinCD({20, 30, 95}) = 2` [because `{20, 30}` and `{95}`]

# Question 1 Solution

1. For any pair of files $f_1$ and $f_2$ in A, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$
2. For any pair of files $f_1$ and $f_2$ in A that belong on a single CD in an optimal solution, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$
3. If $f_1$ and $f_2$ are the largest and smallest in A, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$

Option 3 **does not work** - largest and smallest file may not fit into a single CD

Counterexample: A = `{10, 20, 30, 95, 99}`

Optimal: MinCD(A) = 4 [`{10, 20}, {30}, {95}, {99}` is a possible optimal]

BUT choose largest and smallest pair, e.g. `{10, 99}`

This doesn't even fit!

`MinCD(A) ≠ 1 + MinCD({20, 30, 95})`

`4 ≠ 1 + 2`

`MinCD({20, 30, 95}) = 2` [because `{20, 30}` and `{95}`]

# Question 1 Solution

1. For any pair of files $f_1$ and $f_2$ in A, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$
2. For any pair of files $f_1$ and $f_2$ in A that belong on a single CD in an optimal solution, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$
3. If $f_1$ and $f_2$ are the largest and smallest in A, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$

Option 2 is okay! Consider any optimal solution

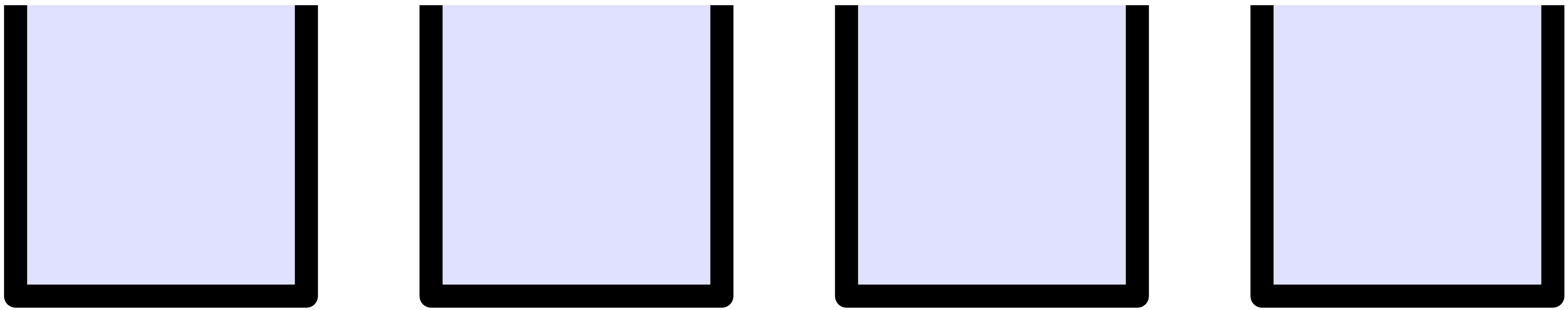
1. Remove any pair that is in the same CD in the optimal solution
2. The rest of the files must be stored optimally
    a. If it is not stored optimally (i.e. it takes up more space), we can always reduce the total number of CDs used by using optimal solution

CDs - Rectangles
Files - Circles

2. For any pair of files $f_1$ and $f_2$ in A that belong on a single CD in an optimal solution, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$

# Q1 - Option 2

Note: Rectangular-shaped CDs

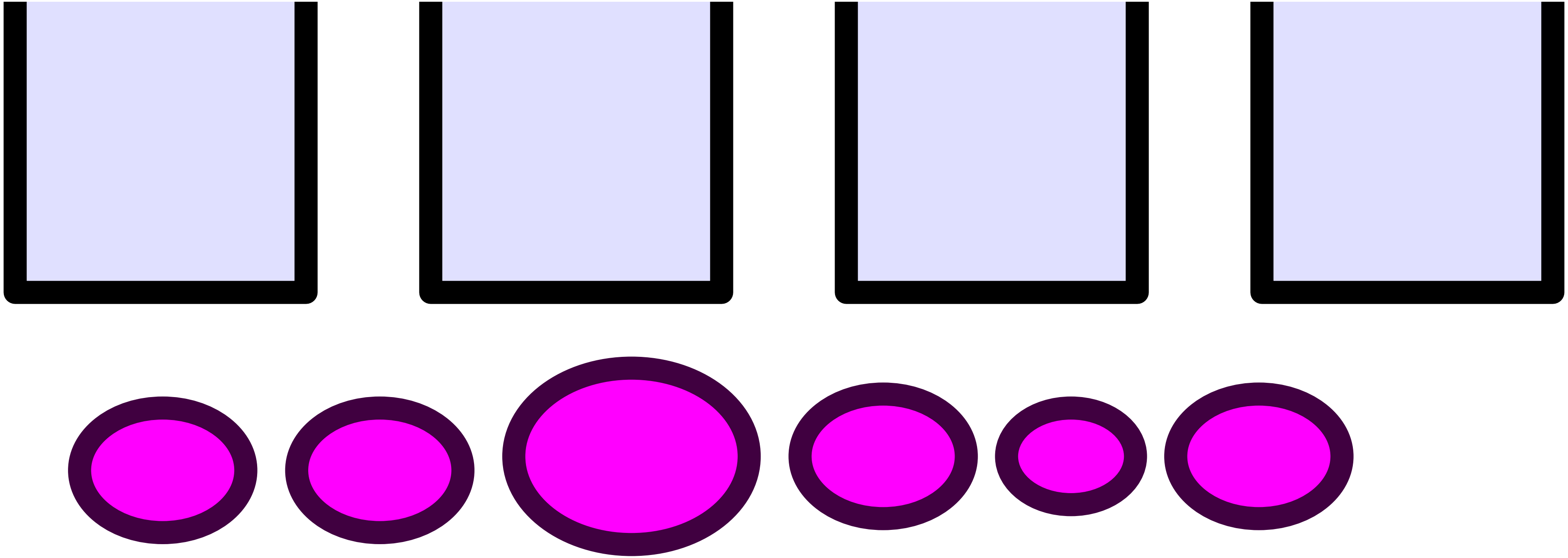


CDs - Rectangles
Files - Circles

2. For any pair of files $f_1$ and $f_2$ in A that belong on a single CD in an optimal solution, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$

# Q1 - Option 2

And circle are Files

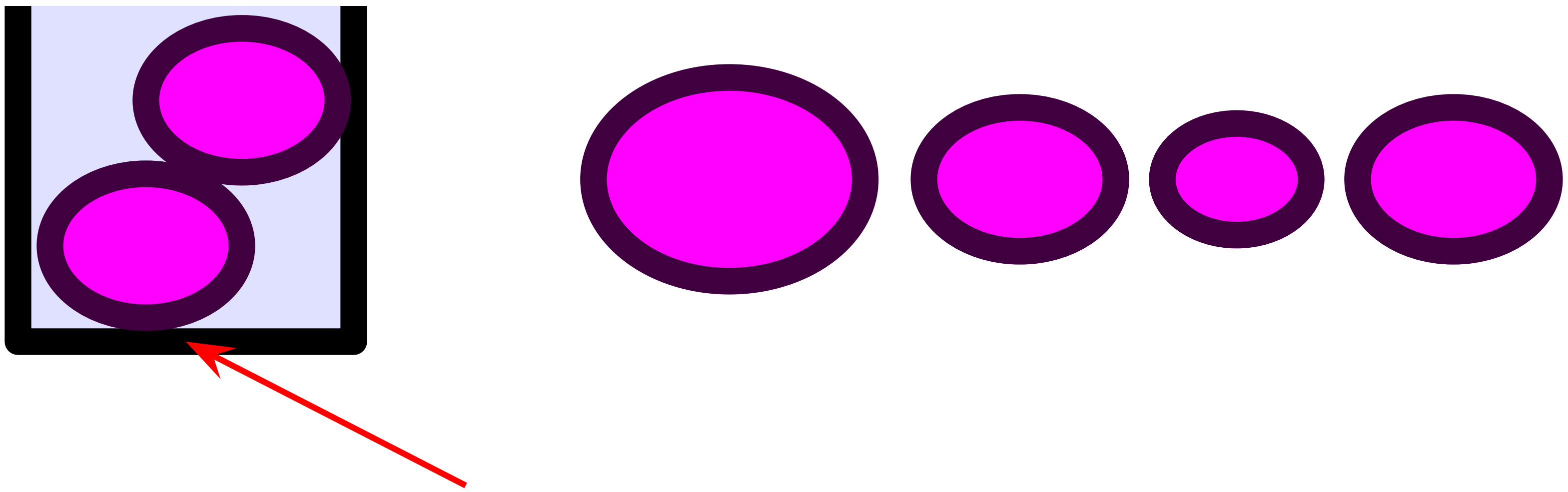


CDs - Rectangles
Files - Circles

2. For any pair of files $f_1$ and $f_2$ in A that belong on a single CD in an optimal solution, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$

## Q1 - Option 2

Assume you can **magically choose** a pair of files in the optimal solutions
(We will show how later)



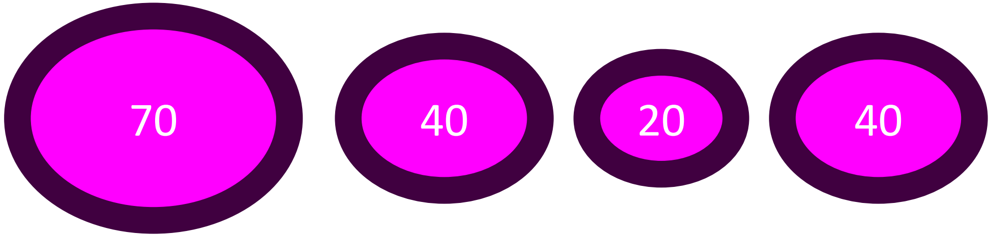
These two are paired together
in an optimal solution (assume we magically know that it is)

CDs - Rectangles
Files - Circles

2. For any pair of files $f_1$ and $f_2$ in A that belong on a single CD in an optimal solution, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$

## Q1 - Option 2

After choosing that pair, we have these subproblems

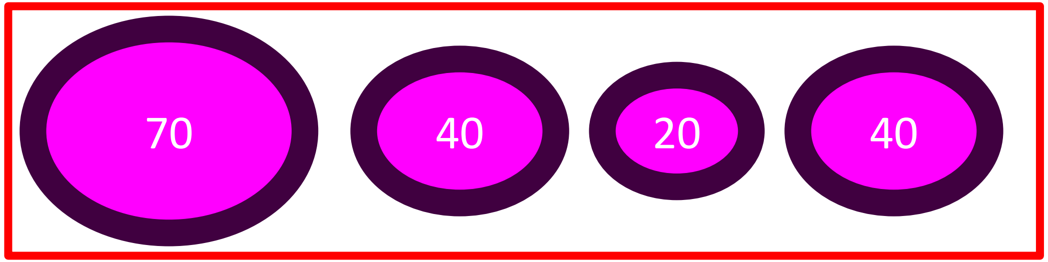


CDs - Rectangles
Files - Circles

2. For any pair of files $f_1$ and $f_2$ in A that belong on a single CD in an optimal solution, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$

## Q1 - Option 2

These remaining problems is solved **optimally too**, to construct our solution



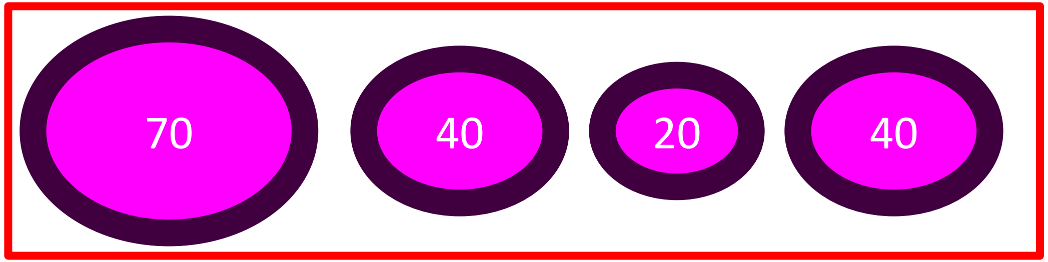

Find optimal ans to this

CDs - Rectangles
Files - Circles

2. For any pair of files $f_1$ and $f_2$ in A that belong on a single CD in an optimal solution, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$

## Q1 - Option 2

Why is the subproblem solved optimally?
Prove by contradiction!



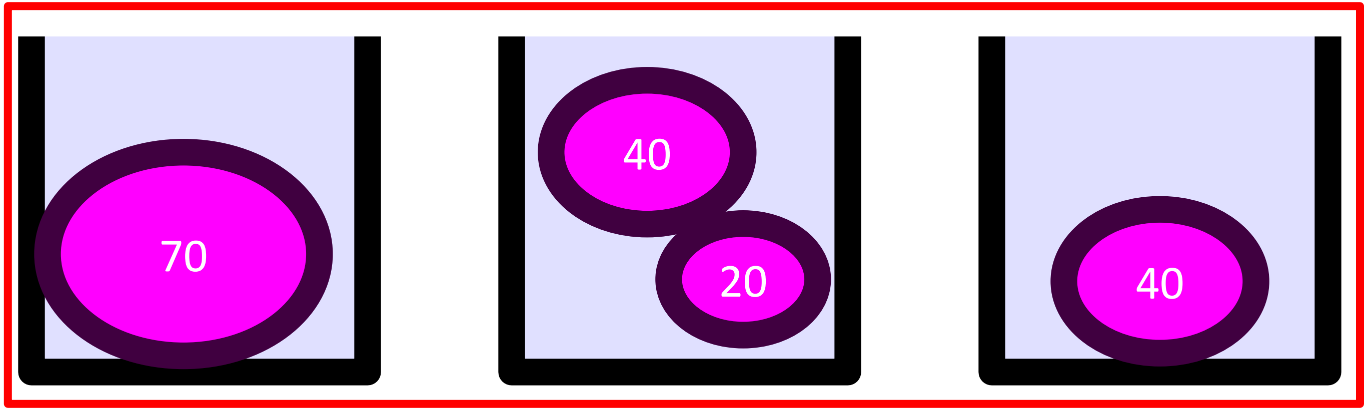

Find optimal ans to this

CDs - Rectangles
Files - Circles

2. For any pair of files $f_1$ and $f_2$ in A that belong on a single CD in an optimal solution, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$

## Q1 - Option 2

Let’s say we can find an optimal solution by using the suboptimal solution of subproblem



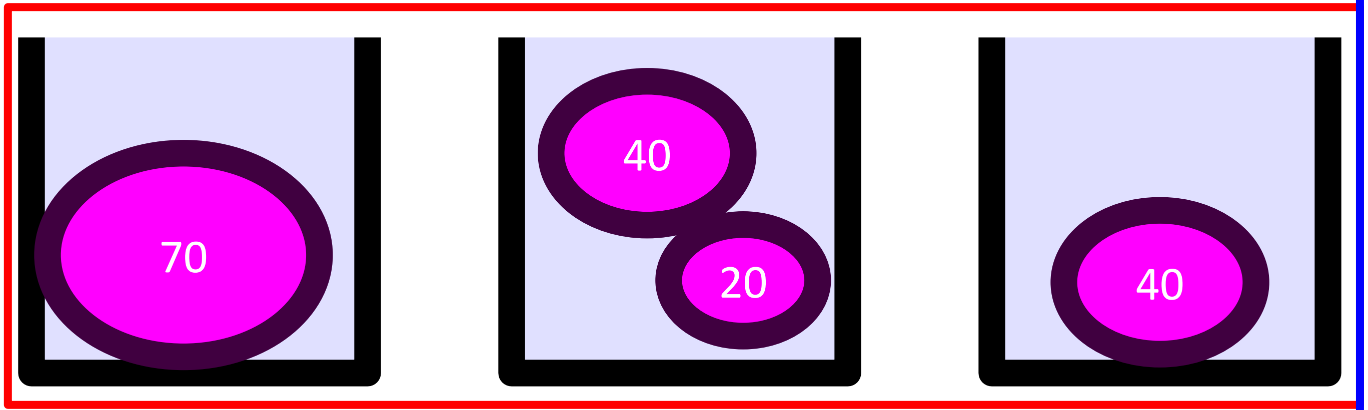

Suboptimal solution to subproblem: 3

CDs - Rectangles
Files - Circles

2. For any pair of files $f_1$ and $f_2$ in A that belong on a single CD in an optimal solution, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$

## Q1 - Option 2

Let's say we can find an optimal solution by using the suboptimal solution of subproblem




Suboptimal solution to subproblem: 3

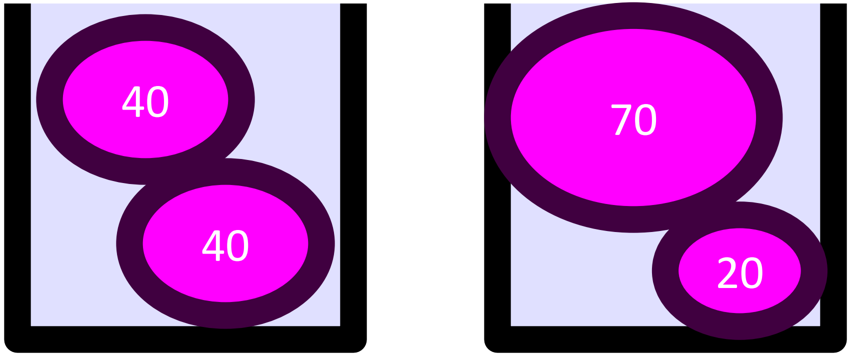
Claim: Optimal solution to whole problem is 1 + 3 = 4
(Using the suboptimal solution)

CDs - Rectangles
Files - Circles

2. For any pair of files $f_1$ and $f_2$ in A that belong on a single CD in an optimal solution, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$

## Q1 - Option 2

Now consider **optimal** solution of subproblem = 2 instead



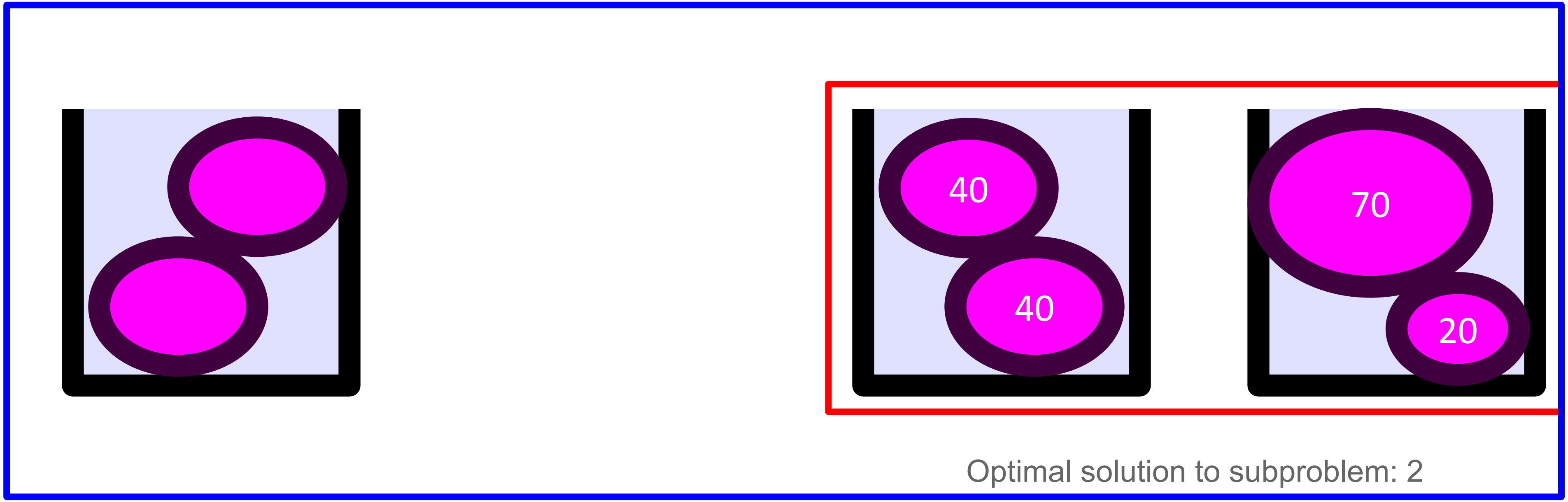

Optimal solution to subproblem: 2

CDs - Rectangles
Files - Circles

2. For any pair of files $f_1$ and $f_2$ in A that belong on a single CD in an optimal solution, $MinCD(A) = 1 + MinCD(A \setminus \{f_1, f_2\})$

## Q1 - Option 2

Then our claimed solution was **NOT** optimal




Claim: Optimal solution to whole problem is 1 + 3 = 4

BUT: we have obtained something even better than optimal: 1 + 2 = 3 -- contradiction!

# Note on the optimal solution

- In the previous example, we gave a concrete example of what an optimal solution looks like
- You don’t actually need to do this. You can argue abstractly, saying something like optimal solution returns value < value returned by suboptimal solution
- I only gave a concrete number for illustration purposes

# Q1 Option 2 - What we just showed

What we just showed is that the problem exhibits optimal substructure (You can reconstruct an optimal solution to the problem, by using optimal solutions to the subproblem)

# Q1 Option 2 - What we just showed

What we just showed is that the problem exhibits optimal substructure (You can reconstruct an optimal solution to the problem, by using optimal solutions to the subproblem)

What we should do:

1. Find out how to choose a pair of file correctly (just now we assumed we can, next qn figures out the algorithm)

# Q1 Option 2 - What we just showed

What we just showed is that the problem exhibits optimal substructure (You can reconstruct an optimal solution to the problem, by using optimal solutions to the subproblem)

What we should do:

1. Find out how to choose a pair of file correctly (just now we assumed we can, next qn figures out the algorithm)
2. Recurse and find optimal solution to subproblem

# Question 2: Making the greedy choice

# Question 2

- Bob has music files that he wants to burn into CDs
- CD storage capacity = 100 MB
- Cannot split music file, i.e., no burning of single file to more than 1 CD
- Not more than two music files per CD

**Assume any optimal solution contains pair burned into a CD**. Select all that is true:

1. The smallest file must be included in a pair in some optimal solution.
2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit onto one CD must be included in a pair in some optimal solution.
3. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file must be included in a pair in some optimal solution.

# Q2 Answer

1. The smallest file must be included in a pair in some optimal solution
2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution
3. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file must be included in a pair in some optimal solution

Answer: Options 1 and 2 only

# Q2 Answer

1. The smallest file must be included in a pair in some optimal solution
2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution
3. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file must be included in a pair in some optimal solution

Answer: Options 1 and 2 only

Option 3 is false, because smallest and largest file might not even fit!

e.g. {20, 90}

# Q2 Answer

1. The smallest file must be included in a pair in some optimal solution
2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution
3. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file must be included in a pair in some optimal solution

Answer: Options 1 and 2 only

Option 1 (Greedy choice):

- Smallest file not included in a pair, swap it with any file included in a pair in an optimal solution
    - Number of CDs do not change

# Q2 Answer

1. The smallest file must be included in a pair in some optimal solution
2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution
3. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file must be included in a pair in some optimal solution

Answer: Options 1 and 2 only

Option 1 (Greedy choice):

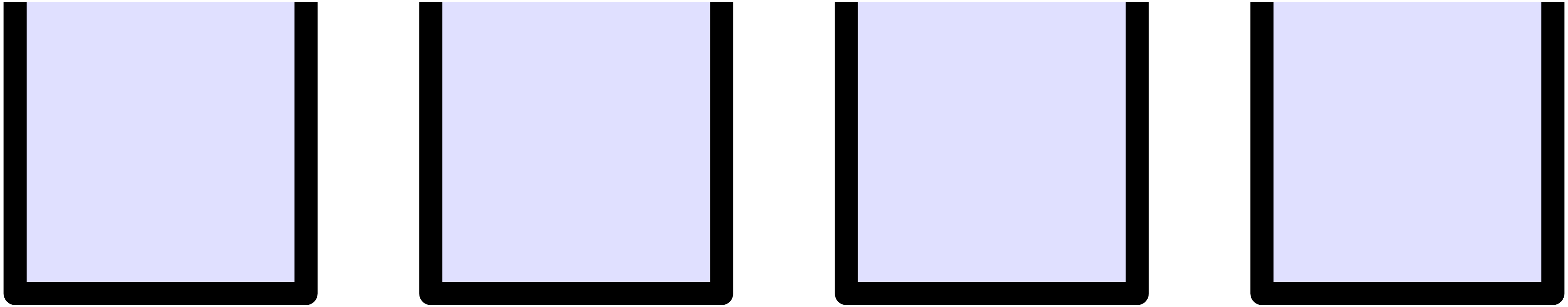
- Smallest file not included in a pair, swap it with any file included in a pair in an optimal solution
  - Number of CDs do not change
- Hence, there exists an optimal solution that contains smallest file in a pair, if an optimal solution with a pair exists

CDs - Rectangles
Files - Circles

1. The smallest file must be included in a pair in some optimal solution

## Q2 - Option 1

Note: Rectangular-shaped CDs

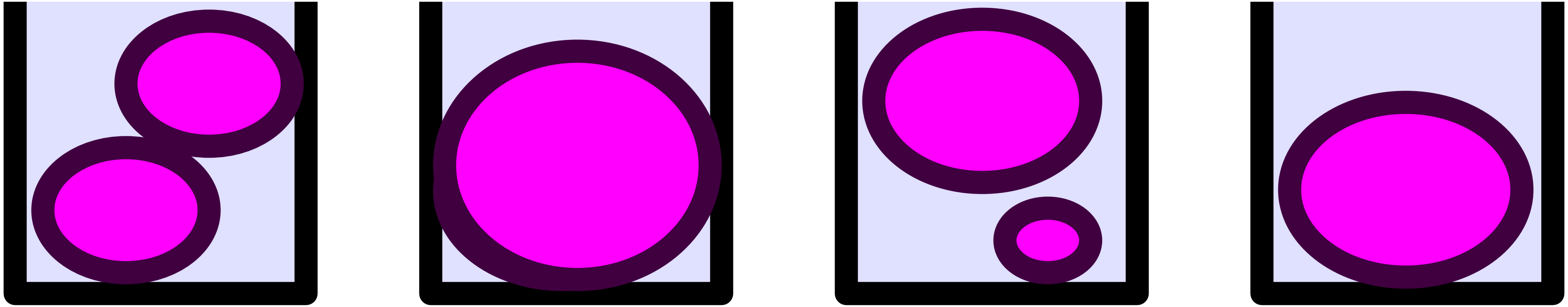


CDs - Rectangles
Files - Circles

1. The smallest file must be included in a pair in some optimal solution

## Q2 - Option 1

`Let’s assume this is **optimal**`



Optimal Pairing

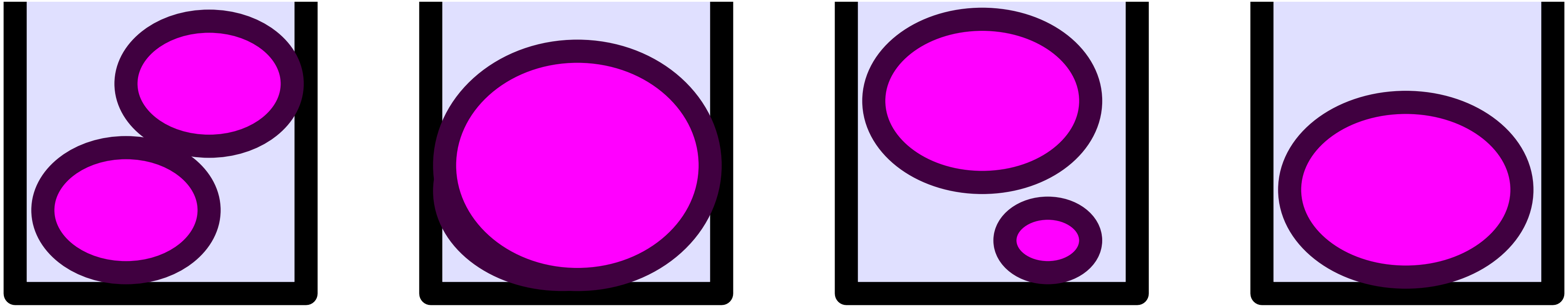
CDs - Rectangles
Files - Circles

1. The smallest file must be included in a pair in some optimal solution

## Q2 - Option 1

Two possibilities - Smallest is:

1. In pair
2. Or not



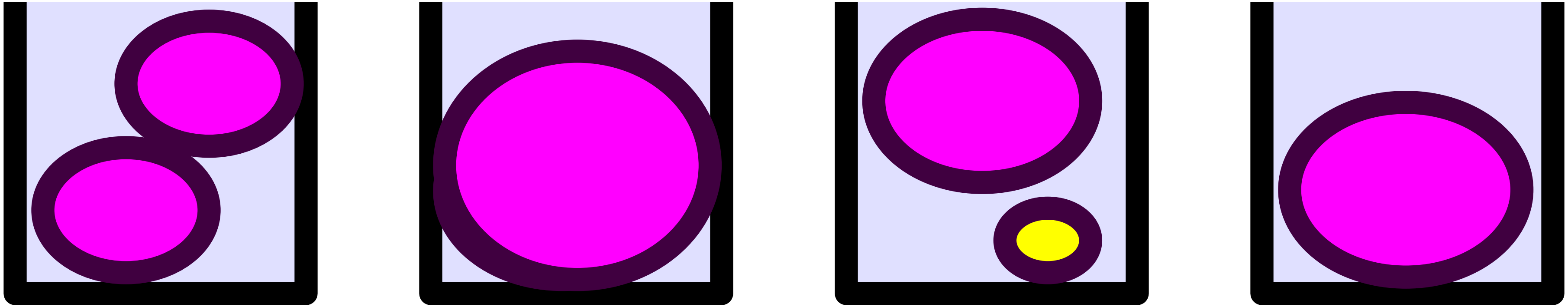
Optimal Pairing

CDs - Rectangles
Files - Circles

1. The smallest file must be included in a pair in some optimal solution

## Q2 - Option 1

If it already is in a pair -- call it a day!



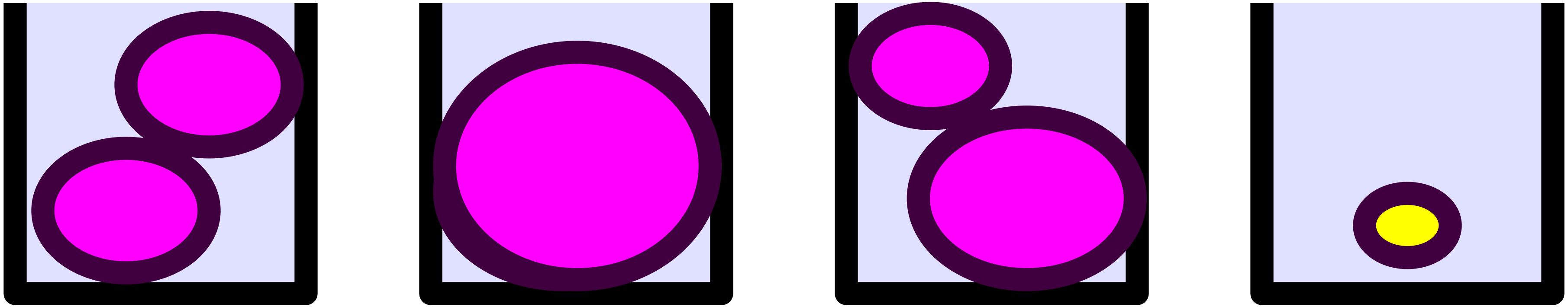
Optimal Pairing

CDs - Rectangles
Files - Circles

1. The smallest file must be included in a pair in some optimal solution

## Q2 - Option 1

Suppose this is an optimal solution where the smallest is NOT in a pair



Note: There are multiple solutions possible that give the same optimal result. Here, we show another arrangement where smallest is not in a pair

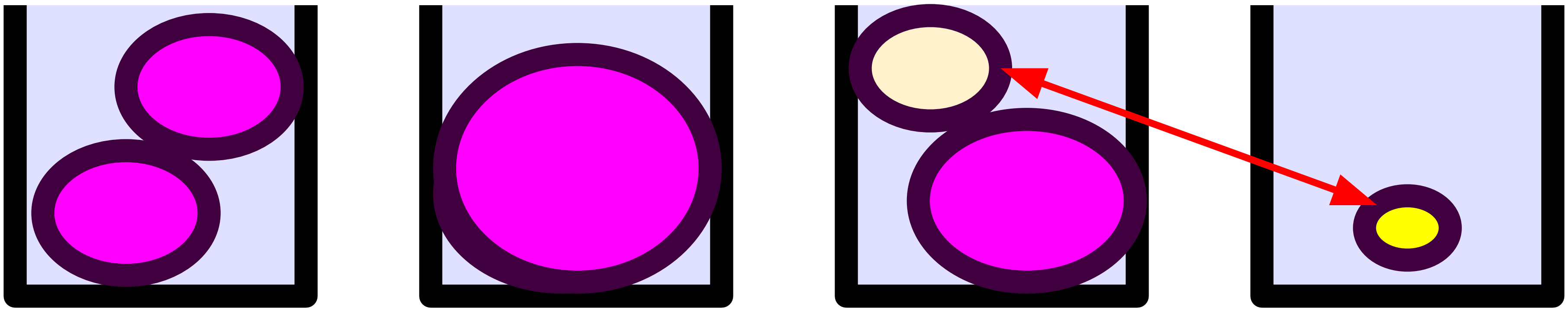
Optimal Pairing

CDs - Rectangles
Files - Circles

1. The smallest file must be included in a pair in some optimal solution

## Q2 - Option 1

Then we can swap smallest into another pair (because it will still fit < 10)



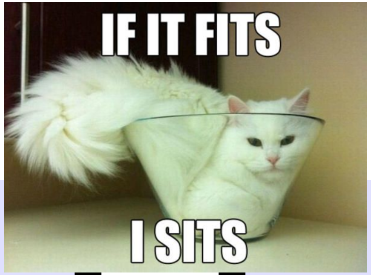
Optimal Pairing

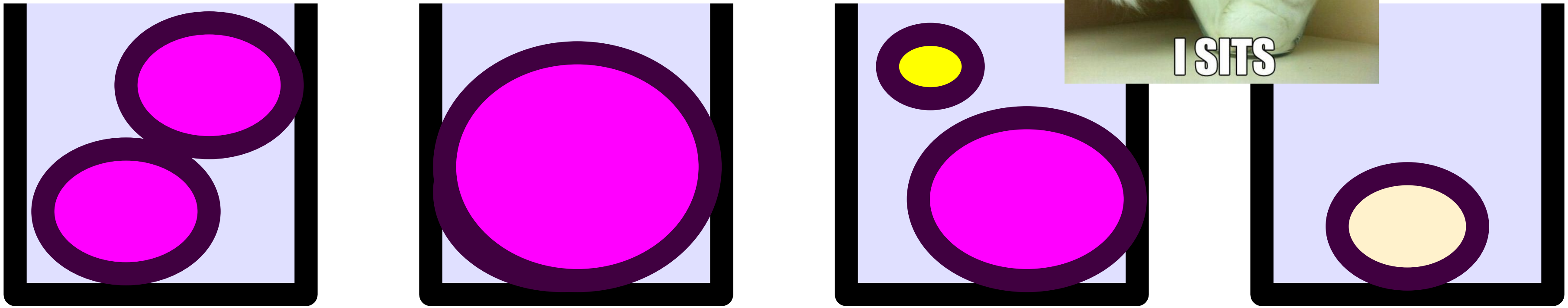
CDs - Rectangles
Files - Circles

1. The smallest file must be included in a pair in some optimal solution

## Q2 - Option 1

This is still optimal! Hence, some optimal solution contains the smallest



Optimal Pairing

# Q2 Answer

1. The smallest file must be included in a pair in some optimal solution
2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution
3. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file must be included in a pair in some optimal solution

Answer: Options 1 and 2 only

Option 1 says: there is an optimal solution where smallest file in a pair, if optimal solution contains a pair

# Q2 Answer

1. The smallest file must be included in a pair in some optimal solution
2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution
3. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file must be included in a pair in some optimal solution

Answer: Options 1 and 2 only

Option 1 says: there is an optimal solution where smallest file in a pair, if optimal solution contains a pair

Option 2 (Greedy choice):

Assume an optimal solution where smallest file is paired with *currP*, but largest that fits is *biggestFit*. Swap *currP* with *biggestFit*.

# Q2 Answer

1. The smallest file must be included in a pair in some optimal solution
2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution
3. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file must be included in a pair in some optimal solution

Answer: Options 1 and 2 only

Option 1 says: there is an optimal solution where smallest file in a pair, if optimal solution contains a pair

Option 2 (Greedy choice):

Assume an optimal solution where smallest file is paired with *currP*, but largest that fits is *biggestFit*. Swap *currP* with *biggestFit*.

- *biggestFit* originally not paired -- still optimal solution

# Q2 Answer

1. The smallest file must be included in a pair in some optimal solution
2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution
3. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file must be included in a pair in some optimal solution

Answer: Options 1 and 2 only

Option 1 says: there is an optimal solution where smallest file in a pair, if optimal solution contains a pair

Option 2 (Greedy choice):

Assume an optimal solution where smallest file is paired with *currP*, but largest that fits is *biggestFit*. Swap *currP* with *biggestFit*.

- *biggestFit* originally not paired -- still optimal solution
- *biggestFit* paired
  - the *currP* coming into the slot of *biggestFit* must fit also (because *biggestFit* is larger)

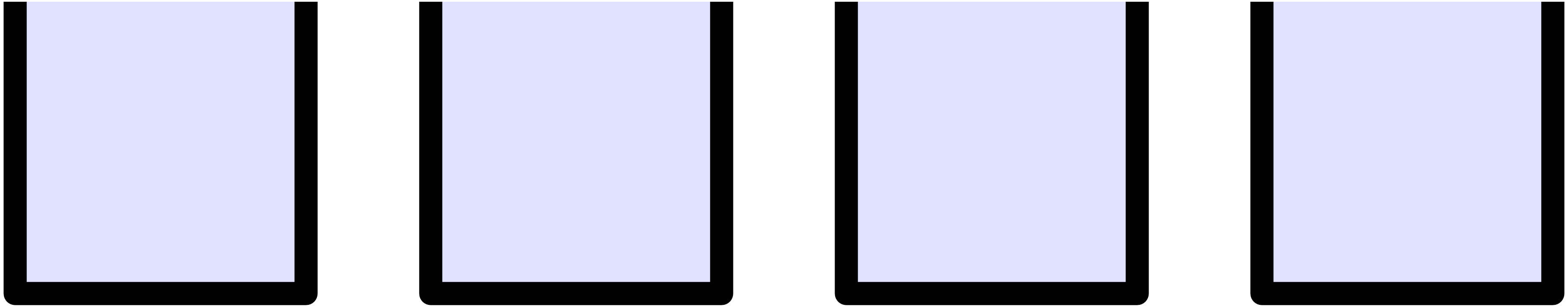
Optimal solution with biggestFit paired with smallest file exists

CDs - Rectangles
Files - Circles

2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution

## Q2 - Option 2

Note: Rectangular-shaped CDs

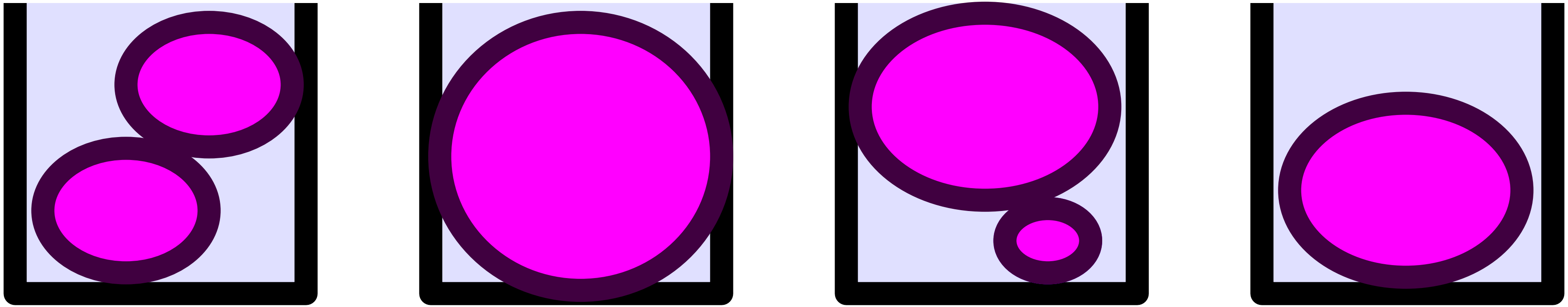


CDs - Rectangles
Files - Circles

2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution

## Q2 - Option 2

Let's assume this is **optimal**



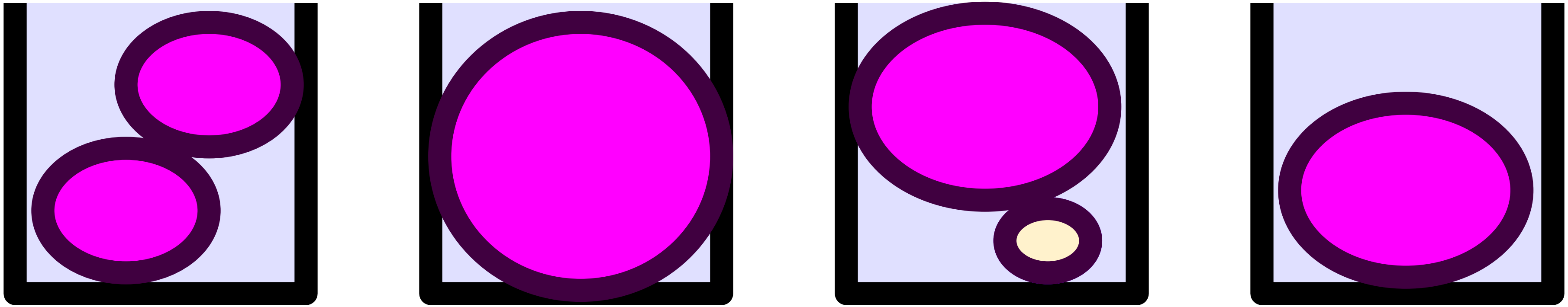
Optimal Pairing

CDs - Rectangles
Files - Circles

2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution

## Q2 - Option 2

Look at the smallest element



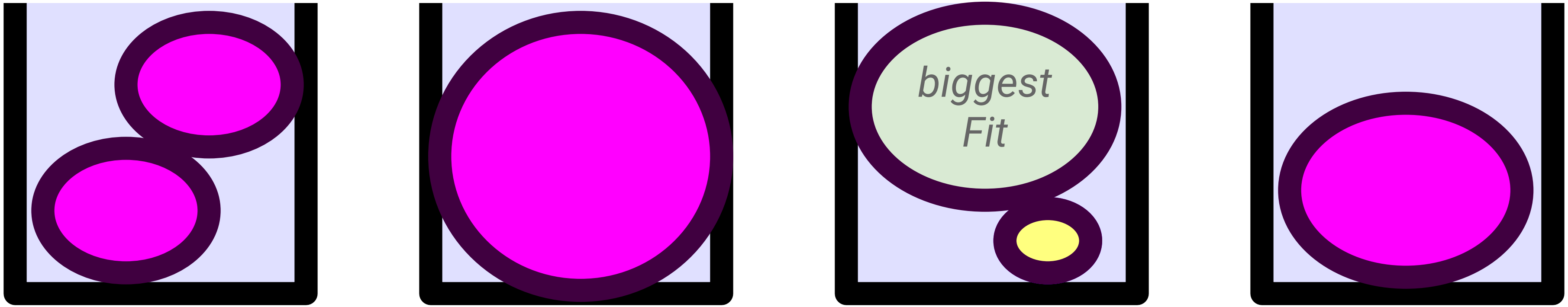
Optimal Pairing

CDs - Rectangles
Files - Circles

2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution

# Q2 - Option 2

**Look at the biggest element it can fit with**




Optimal Pairing

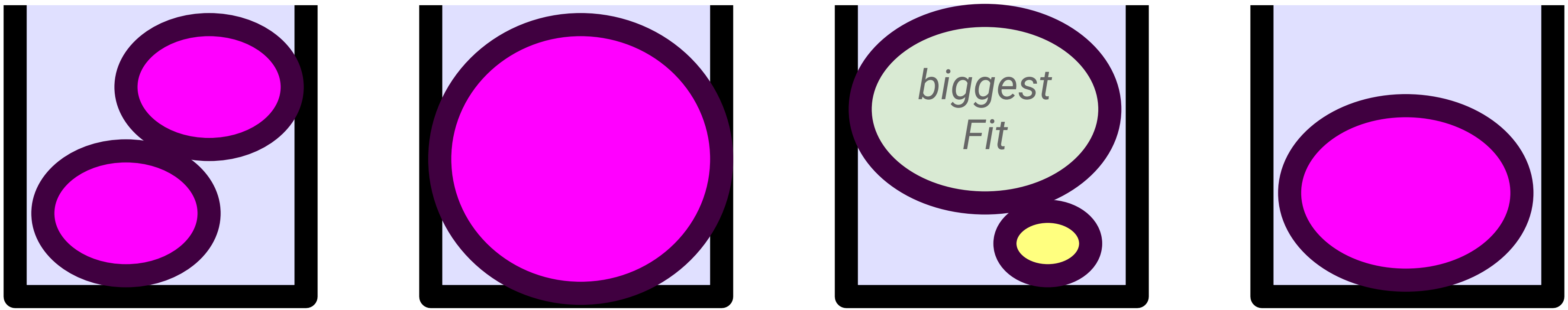
CDs - Rectangles
Files - Circles

2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution

## Q2 - Option 2

Two possibilities -

1. they are already a pair
2. or not



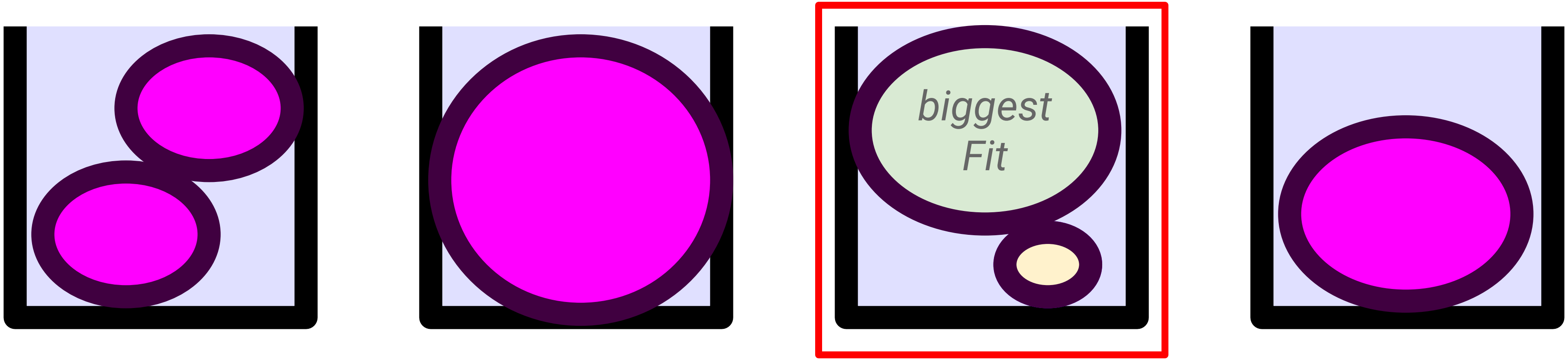

Optimal Pairing

CDs - Rectangles
Files - Circles

2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution

## Q2 - Option 2

If they already are, we are done!




Already a pair!

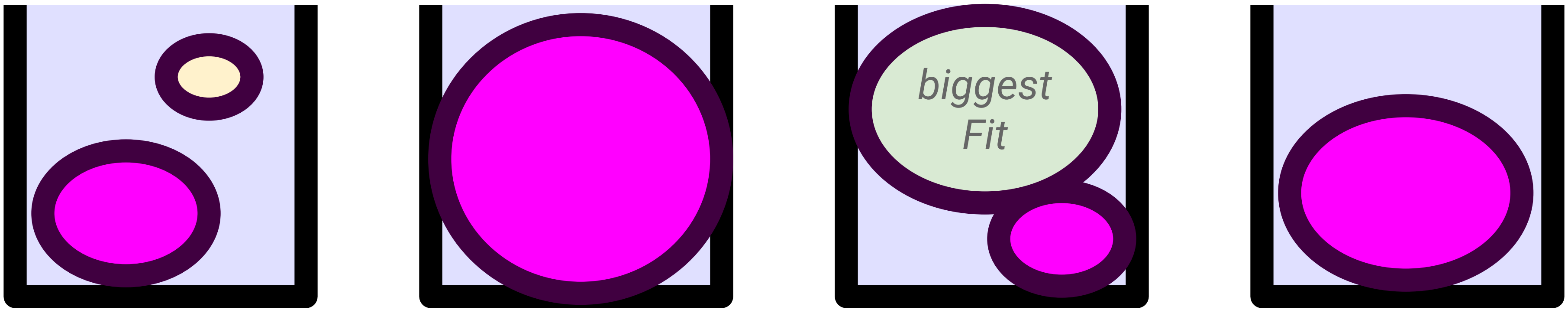
Optimal Pairing

CDs - Rectangles
Files - Circles

2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution

## Q2 - Option 2

Here is **another** optimal answer, where the two are not a pair already



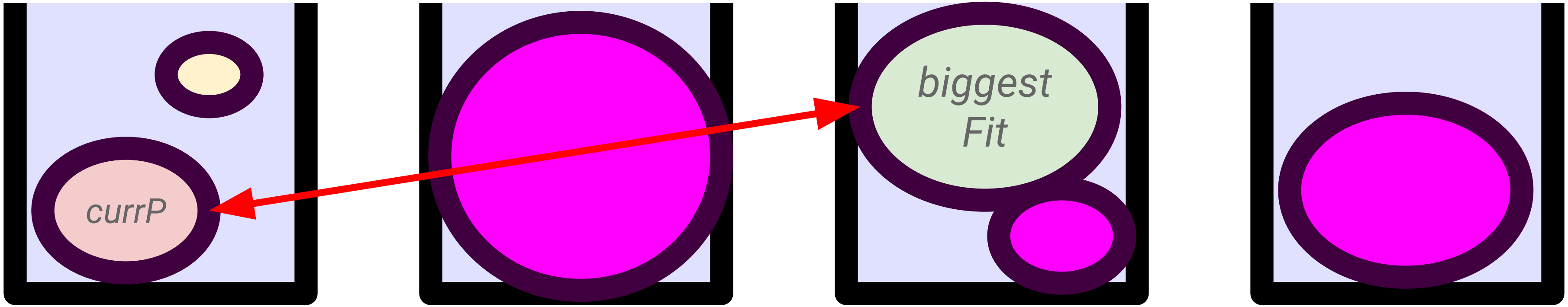

Optimal Pairing

CDs - Rectangles
Files - Circles

2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution

## Q2 - Option 2

Then swap these two!



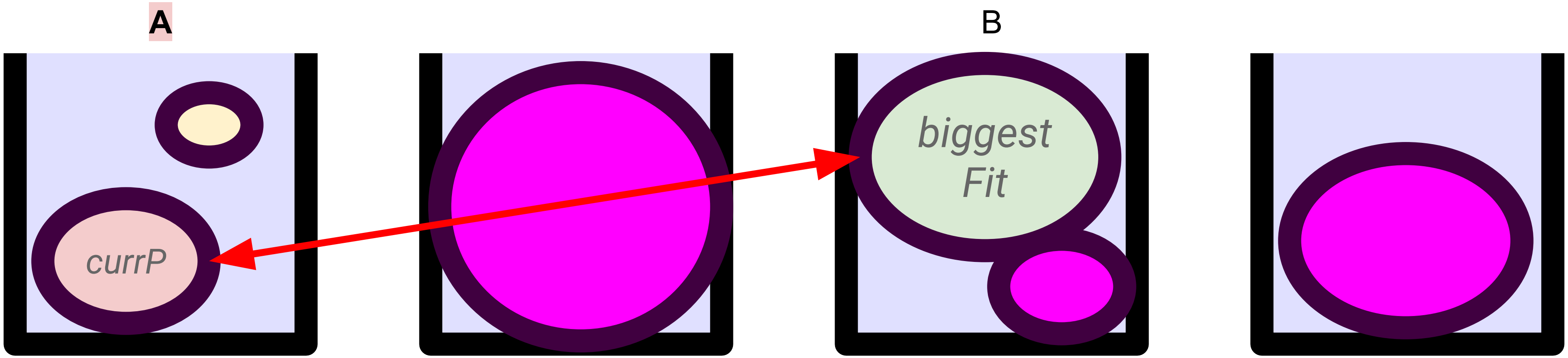

Optimal Pairing

CDs - Rectangles
Files - Circles

2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution

## Q2 - Option 2

Then swap these two!




Why is this ok?

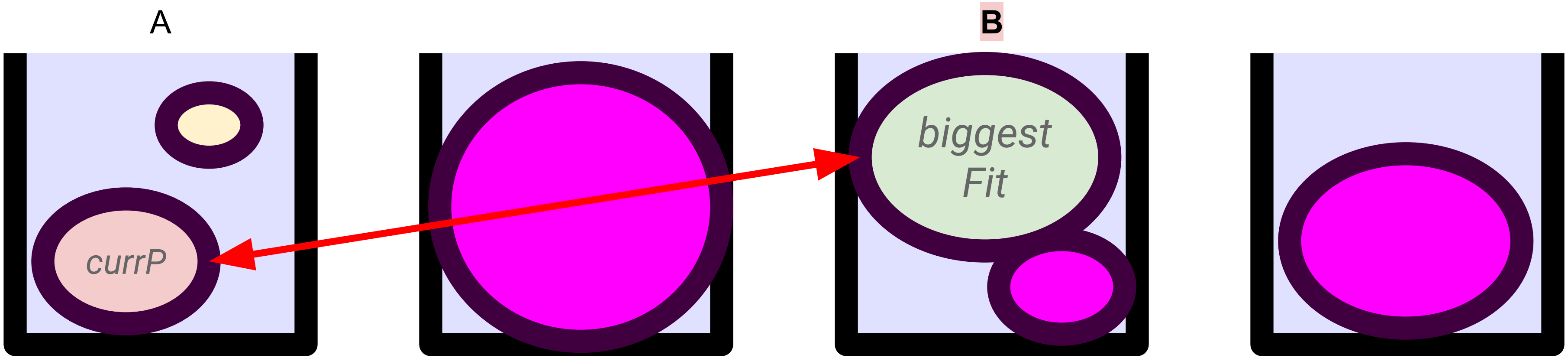
- By construction, *biggestFit* can fit with smallest element
  - It must be able to fit in CD **A**

CDs - Rectangles
Files - Circles

2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution

## Q2 - Option 2

```
Then swap these two!
```




Why is this ok?

- By construction, *biggestFit* can fit with smallest element
  - It must be able to fit in CD A
- *currP* is smaller than *biggestFit* (since *biggestFit* is well.. the biggest that can fit)
  - It must be able to fit in CD **B**

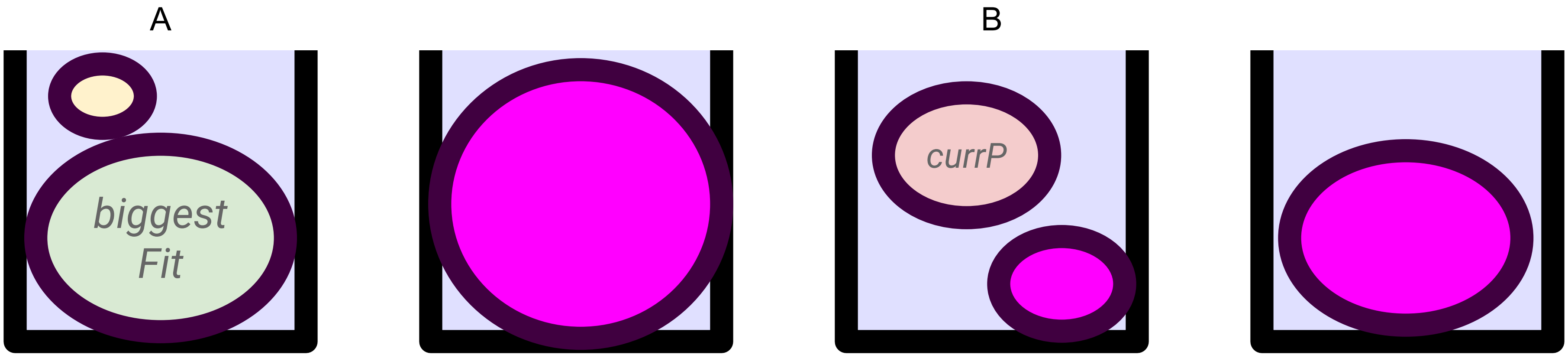
Optimal Pairing

CDs - Rectangles
Files - Circles

2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution

## Q2 - Option 2

```
No new CDs are created -- still optimal!
```




Optimal Pairing

# Q2 - What we just showed

We now know a way to “make a choice” for the current subproblem

- Look for the smallest element, and pair it with the largest fitting one

# Q2 - What we just showed

We now know a way to “make a choice” for the current subproblem

- Look for the smallest element, and pair it with the largest fitting one
- This question tells us that using such a pair can still give us an optimal solution -- this greedy choice won’t “hurt” our result

# Question 3: Deriving the greedy algorithm

# Question 3

- Bob has music files that he wants to burn into CDs
- CD storage capacity = 100 MB
- Cannot split music file, i.e., no burning of single file to more than 1 CD
- Not more than two music files per CD
- Given set A of file sizes, each smaller than 100MB, let `MinCD(A)` denote the minimum number of CDs required to fit the files described in A

**Derive an algorithm to compute `MinCD(A)`**

Let `filesizes =`

`[89, 59, 32, 74, 81, 12, 7, 49, 43, 51, 61, 91, 27]`

What is the value of `MinCD(filesizes)`?

1. The smallest file must be included in a pair in some optimal solution
2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution

# Question 3 solution

Idea: We want to greedily find the first pair (smallest element, with the largest that fits with it)

1. The smallest file must be included in a pair in some optimal solution
2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution

# Question 3 solution

Idea: We want to greedily find the first pair (smallest element, with the largest that fits with it)

1. Sort the array

# Question 3 solution

1. The smallest file must be included in a pair in some optimal solution
2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution

Idea: We want to greedily find the first pair (smallest element, with the largest that fits with it)

1. Sort the array
2. Maintain two pointers:
   a. Going left to right (to keep taking the smallest element **greedily**)
   b. Going right to left (keep finding the largest element that can fit)

# Question 3 solution

1. The smallest file must be included in a pair in some optimal solution
2. The pair $\{f_1, f_2\}$ where $f_1$ is the smallest file and $f_2$ is the largest file such that $f_1$ and $f_2$ fit into one CD must be included in a pair in some optimal solution

Idea: We want to greedily find the first pair (smallest element, with the largest that fits with it)

1. Sort the array
2. Maintain two pointers:
   a. Going left to right (to keep taking the smallest element **greedily**)
   b. Going right to left (keep finding the largest element that can fit)
3. Collect the remaining elements that were “skipped”

# Question 3 solution

[7, 12, 27, 32, 43, 49, 51, 59, 62, 74, 81, 89, 91]

L R

Run the greedy algorithm to get

{7,91}, {12,81}, {27, 62}, {32, 59}, {43, 51}, {49}, {74}, {89}

1. Sort the array
2. Maintain two pointers:
   a. Going left to right (to keep taking the smallest element greedily)
   b. Going right to left (keep finding the largest element that can fit)
3. Collect the remaining elements that were “skipped”

# Question 3 solution

[**7**, 12, 27, 32, 43, 49, 51, 59, 62, 74, 81, 89, **91**]

L R

Run the greedy algorithm to get

**{7,91}**, {12,81}, {27, 62}, {32, 59}, {43, 51}, {49}, {74}, {89}

1. Sort the array
2. Maintain two pointers:
   a. Going left to right (to keep taking the smallest element greedily)
   b. Going right to left (keep finding the largest element that can fit)
3. Collect the remaining elements that were “skipped”

# Question 3 solution

[**7**, **12**, 27, 32, 43, 49, 51, 59, 62, 74, 81, 89, **91**]

L R

Run the greedy algorithm to get

**{7,91}**, {**12**,81}, {27, 62}, {32, 59}, {43, 51}, {49}, {74}, {89}

1. Sort the array
2. Maintain two pointers:
   a. Going left to right (to keep taking the smallest element greedily)
   b. Going right to left (keep finding the largest element that can fit)
3. Collect the remaining elements that were “skipped”

# Question 3 solution

[**7**, **12**, 27, 32, 43, 49, 51, 59, 62, 74, **81**, 89, **91**]

L R

Run the greedy algorithm to get

**{7,91}**, **{12,81}**, {27, 62}, {32, 59}, {43, 51}, {49}, {74}, {89}

1. Sort the array
2. Maintain two pointers:
   a. Going left to right (to keep taking the smallest element greedily)
   b. Going right to left (keep finding the largest element that can fit)
3. Collect the remaining elements that were “skipped”

# Question 3 solution

[7, **12**, **27**, 32, 43, 49, 51, 59, 62, 74, **81**, 89, **91**]

L R

Run the greedy algorithm to get

**{7,91}**, **{12,81}**, {**27**, 62}, {32, 59}, {43, 51}, {49}, {74}, {89}

1. Sort the array
2. Maintain two pointers:
    a. Going left to right (to keep taking the smallest element greedily)
    b. Going right to left (keep finding the largest element that can fit)
3. Collect the remaining elements that were “skipped”

# Question 3 solution

[**7**, **12**, **27**, 32, 43, 49, 51, 59, **62**, 74, **81**, 89, **91**]

L R

Run the greedy algorithm to get

**{7,91}**, **{12,81}**, **{27, 62}**, {32, 59}, {43, 51}, {49}, {74}, {89}

1. Sort the array
2. Maintain two pointers:
   a. Going left to right (to keep taking the smallest element greedily)
   b. Going right to left (keep finding the largest element that can fit)
3. Collect the remaining elements that were “skipped”

# Question 3 solution

[**7**, **12**, **27**, **32**, 43, 49, 51, 59, **62**, 74, **81**, 89, **91**]

L R

Run the greedy algorithm to get

**{7,91}**, **{12,81}**, **{27, 62}**, {**32**, 59}, {43, 51}, {49}, {74}, {89}

1. Sort the array
2. Maintain two pointers:
    a. Going left to right (to keep taking the smallest element greedily)
    b. Going right to left (keep finding the largest element that can fit)
3. Collect the remaining elements that were “skipped”

# Question 3 solution

[7, **12**, **27**, **32**, 43, 49, 51, **59**, **62**, 74, **81**, 89, **91**]

L R

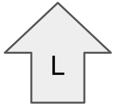
Run the greedy algorithm to get

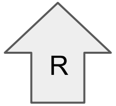
**{7,91}**, **{12,81}**, **{27, 62}**, **{32, 59}**, {43, 51}, {49}, {74}, {89}

1. Sort the array
2. Maintain two pointers:
   a. Going left to right (to keep taking the smallest element greedily)
   b. Going right to left (keep finding the largest element that can fit)
3. Collect the remaining elements that were “skipped”

# Question 3 solution

[7, **12**, **27**, **32**, **43**, 49, 51, **59**, **62**, 74, **81**, 89, **91**]




Run the greedy algorithm to get

**{7,91}**, **{12,81}**, **{27, 62}**, **{32, 59}**, {**43**, 51}, {49}, {74}, {89}

1. Sort the array
2. Maintain two pointers:
   a. Going left to right (to keep taking the smallest element greedily)
   b. Going right to left (keep finding the largest element that can fit)
3. Collect the remaining elements that were “skipped”

# Question 3 solution

[**7**, **12**, **27**, **32**, **43**, 49, **51**, **59**, **62**, 74, **81**, 89, **91**]

L R

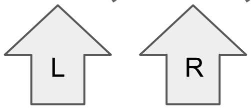
Run the greedy algorithm to get

**{7,91}**, **{12,81}**, **{27, 62}**, **{32, 59}**, **{43, 51}**, {49}, {74}, {89}

1. Sort the array
2. Maintain two pointers:
   a. Going left to right (to keep taking the smallest element greedily)
   b. Going right to left (keep finding the largest element that can fit)
3. Collect the remaining elements that were “skipped”

# Question 3 solution

[7, **12**, **27**, **32**, **43**, **49**, **51**, **59**, **62**, 74, **81**, 89, **91**]




Run the greedy algorithm to get

**{7,91}**, **{12,81}**, **{27, 62}**, **{32, 59}**, **{43, 51}**, **{49}**, {74}, {89}

1. Sort the array
2. Maintain two pointers:
   a. Going left to right (to keep taking the smallest element greedily)
   b. Going right to left (keep finding the largest element that can fit)
3. Collect the remaining elements that were “skipped”

# Question 3 solution

[7, **12**, **27**, **32**, **43**, **49**, **51**, **59**, **62**, **74**, **81**, 89, **91**]


Run the greedy algorithm to get

**{7,91}**, **{12,81}**, **{27, 62}**, **{32, 59}**, **{43, 51}**, **{49}**, **{74}**, {89}

1. Sort the array
2. Maintain two pointers:
   a. Going left to right (to keep taking the smallest element greedily)
   b. Going right to left (keep finding the largest element that can fit)
3. Collect the remaining elements that were “skipped”

# Question 3 solution

**[7, 12, 27, 32, 43, 49, 51, 59, 62, 74, 81, 89, 91]**

R L

Run the greedy algorithm to get

**{7,91}, {12,81}, {27, 62}, {32, 59}, {43, 51}, {49}, {74}, {89}**

1. Sort the array
2. Maintain two pointers:
   a. Going left to right (to keep taking the smallest element greedily)
   b. Going right to left (keep finding the largest element that can fit)
3. Collect the remaining elements that were “skipped”

# Question 3 solution

[7, **12**, **27**, **32**, **43**, **49**, **51**, **59**, **62**, **74**, **81**, **89**, **91**]

Run the greedy algorithm to get
**{7,91}**, **{12,81}**, **{27, 62}**, **{32, 59}**, **{43, 51}**, **{49}**,
**{74}**, **{89}**

**Total 8 CDs required!**

# Question 3 solution

**[7, 12, 27, 32, 43, 49, 51, 59, 62, 74, 81, 89, 91]**

Run the greedy algorithm to get
**{7,91}, {12,81}, {27, 62}, {32, 59}, {43, 51}, {49}, {74}, {89}**

**Total 8 CDs required!**

Time Complexity:

- O(nlogn) for sorting
- O(n) for pointers traversal

# Question 4: Activity Selection

# Activity Selection Problem

Given a set of activities $S = \{a_1, a_2, \ldots, a_n\}$:

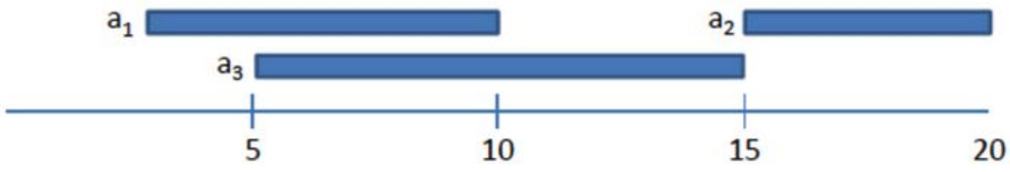
- Each activity takes place during $[s_i, f_i)$
- Two activities $a_i$ and $a_j$ are **compatible** if their time intervals don't overlap: $s_i \geq f_j$ or $s_j \geq f_i$.

**Problem**: Find a largest subset of mutually compatible activities.

# Activity Selection Problem

**Example:** $a_1$=[3, 10), $a_2$=[15, 20), $a_3$=[5, 15)

- {$a_1$ and $a_2$} and {$a_2$ and $a_3$} are compatible
- {$a_1$ and $a_3$} are not compatible



# Question 4

Which of these greedy strategies work for the activity selection problem?

1. Choose the activity $a$ that **starts last**, discard those that conflict with $a$, and recurse.

2. Choose the activity $a$ that **ends last**, discard those that conflict with $a$, and recurse.

3. Choose the **shortest activity** $a$, discard those that conflict with $a$, and recurse.

# Q4: Optimal Substructure

Suppose an optimal scheduling $S$ contains activity $a_j$.
Let:

$$\mathrm{Before}_j = \{\boldsymbol{a}_i\colon \boldsymbol{f}_i \leq \boldsymbol{s}_j\}$$
$$\mathrm{After}_j = \{\boldsymbol{a}_i\colon \boldsymbol{s}_i \geq \boldsymbol{f}_j\}$$

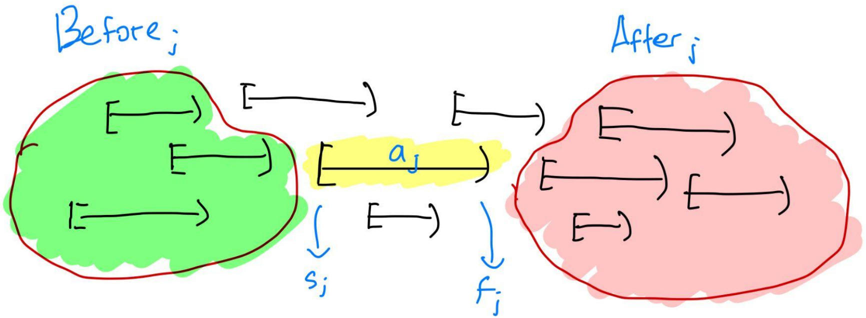
Then, $S$ also contains an optimal scheduling for $\mathrm{Before}_j$ and an optimal scheduling for $\mathrm{After}_j$.

# Q4: Optimal Substructure

Suppose an optimal scheduling $S$ contains activity $a_j$.
Let:

$$\text{Before}_j = \{a_i : f_i \le s_j\}$$
$$\text{After}_j = \{a_i : s_i \ge f_j\}$$

Then, $S$ also contains an optimal scheduling for $\text{Before}_j$ and an optimal scheduling for $\text{After}_j$.

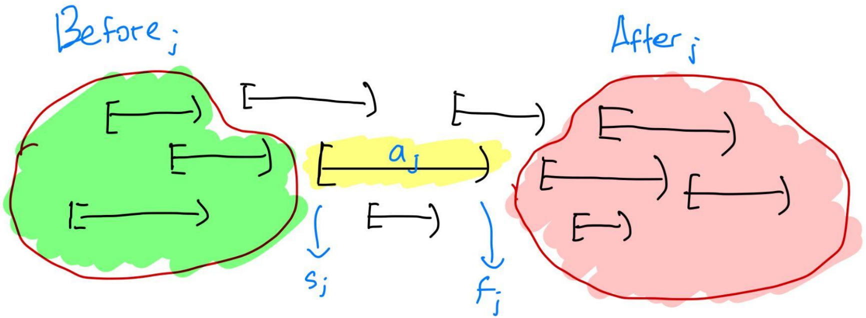


# Q4: Optimal Substructure

Suppose an optimal scheduling $S$ contains activity $a_j$.
Let:

$$\mathrm{Before}_j = \{a_i : f_i \leq s_j\}$$
$$\mathrm{After}_j = \{a_i : s_i \geq f_j\}$$

Then, $S$ also contains an optimal scheduling for $\mathrm{Before}_j$ and an optimal scheduling for $\mathrm{After}_j$.




You can use the same argument like usual to show optimal substructure

# Q4: Intuition

1. Choose the activity that **starts last**, discard those in conflict and recurse
2. Choose the activity that **ends last**, discard those in conflict and recurse
3. Choose the activity that **is the shortest**, discard those in conflict and recurse

If you want to fit as many lessons as possible, which timing would you choose?

# Q4: Intuition

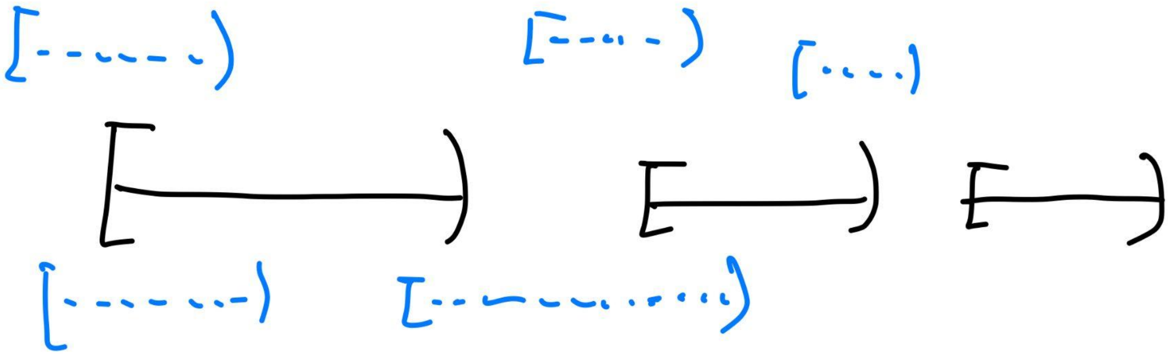
1. Choose the activity that **starts last**, discard those in conflict and recurse
2. Choose the activity that **ends last**, discard those in conflict and recurse
3. Choose the activity that **is the shortest**, discard those in conflict and recurse

If you want to fit as many lessons as possible, which timing would you choose?

Choose the one that starts last - because it “frees up” your day as much as possible

Want to prove: that we can always get an optimal answer by making the greedy choice

# Q4: Proof for greedy choice

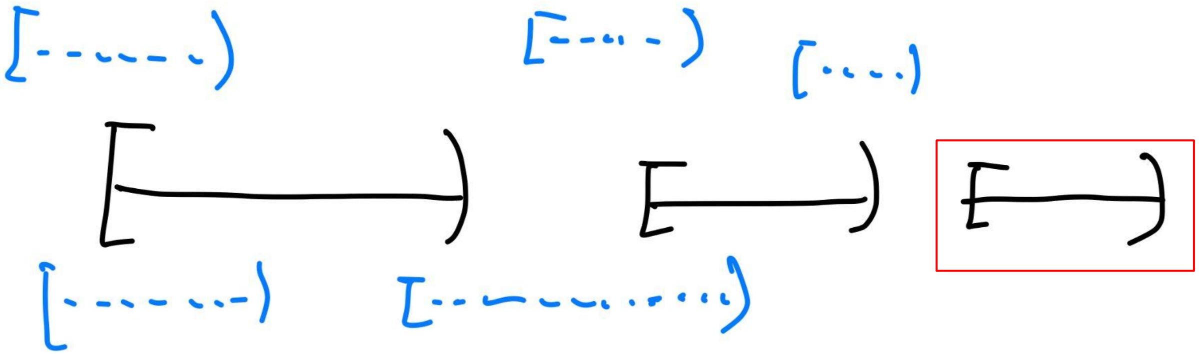


Assume the ones in the black are the optimal solution
Blue ones are removed

Want to prove: that we can always get an optimal answer by making the greedy choice

# Q4: Proof for greedy choice

**Case 1**



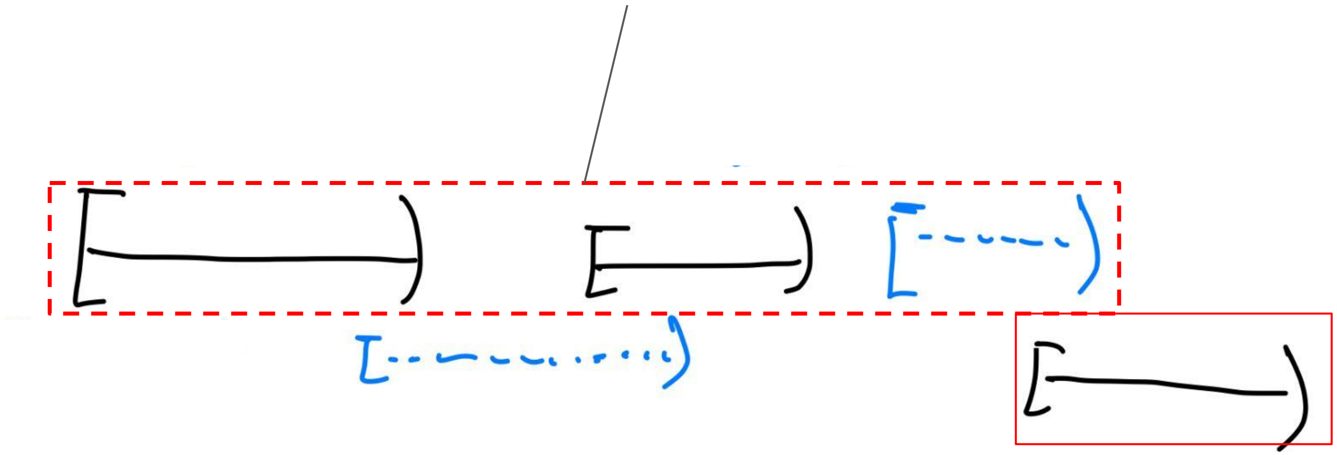
If the start last in optimal solution = start last in the **input**, then ok!

Want to prove: that we can always get an optimal answer by making the greedy choice

# Q4: Proof for greedy choice

**Case 2**

We initially say this was optimal



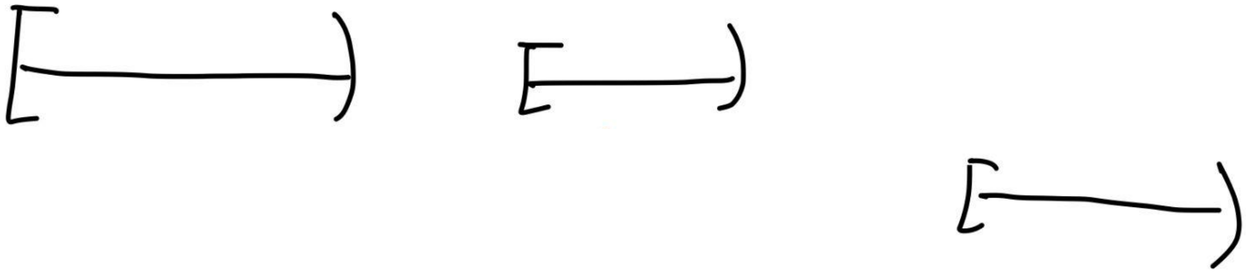
If start last in optimal ≠ start last in input, we can always swap the one from the input into the optimal solution

Note: In case 1, we are saying our optimal solution contains the interval that **actually starts last**.
In case 2, we are saying we managed to come up with an optimal solution that **didn't** need to use the one that starts last (diagram shows another input)

Want to prove: that we can always get an optimal answer by making the greedy choice

# Q4: Proof for greedy choice

**Case 2**



This new one is **still an optimal solution!**

# Q4: What we just proved

By making the greedy choice of always taking the one that **starts last**, our answer is **not** going to get any worse!

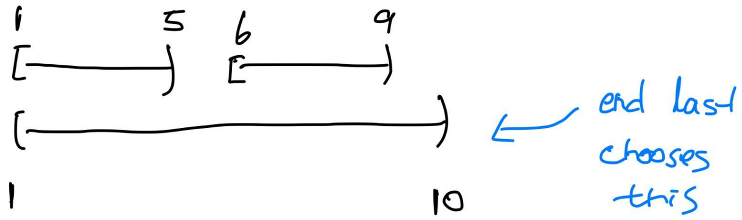
i.e. There is still an optimal solution that has the one that **starts last**

# Q4: Others

1. Choose the activity that **starts last**, discard those in conflict and recurse
2. Choose the activity that **ends last**, discard those in conflict and recurse
3. Choose the activity that **is the shortest**, discard those in conflict and recurse

Option 1: This one works, as proven

Option 2: Does not work:

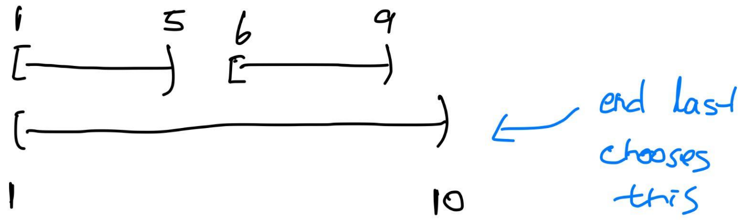


# Q4: Others

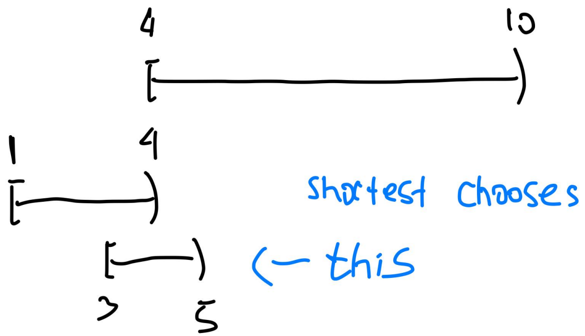
1. Choose the activity that **starts last**, discard those in conflict and recurse
2. Choose the activity that **ends last**, discard those in conflict and recurse
3. Choose the activity that **is the shortest**, discard those in conflict and recurse

Option 1: This one works, as proven

Option 2: Does not work:




Option 3: Does not work:



# Important thing to do in Greedy Algorithm

- In DP: try **all** the subproblems
- In Greedy: try **one** subproblem, chosen **greedily** (usually, something like the one that gives the max/min value)

- You have to prove that the greedy choice will still give you the best optimal solution
  - Usually by the ‘cut-and-paste’ argument
  - ‘Cut’ out the current optimal solution and ‘paste’ a solution using greedy choice
    - Show that the solution stays as “good” (doesn’t become “worse”)