Analysis and Design of Algorithms

Algorithms
CS3230

Tutorial

Week 10

Question 1

Suppose Bob has a collection of music files that he wants to burn into CDs. Each CD has a storage capacity of 100MB. In addition, Bob does not want to store more than two music files per CD. Note that each music file cannot be split and hence cannot be burned on more than 1 CDs. Given a 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 of the following correctly describes an optimal substructure property of the problem, assuming that **at least one pair of files fit onto a CD**?

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

Answer: B

Answer A does not work as we cannot simply pair up any two file and still be optimal.

Counterexample to A:

Consider {10, 20, 80, 90}. Optimal solution is {10,90}, {20, 80}.

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

after pairing {10,20} as {80,90} requires 2 CDs

Question 1

Answer: B

Answer C does not work as the largest and smallest file may not fit into a single CD.

Counterexample to MinCD(A)=1+MinCD(A – {f_1,f_2}) for largest and smallest files f_1, f_2:

Consider {10, 20, 30, 95, 99}. The optimal solution is {10, 20}, {30}, {95}, {99}: 4 CDs.

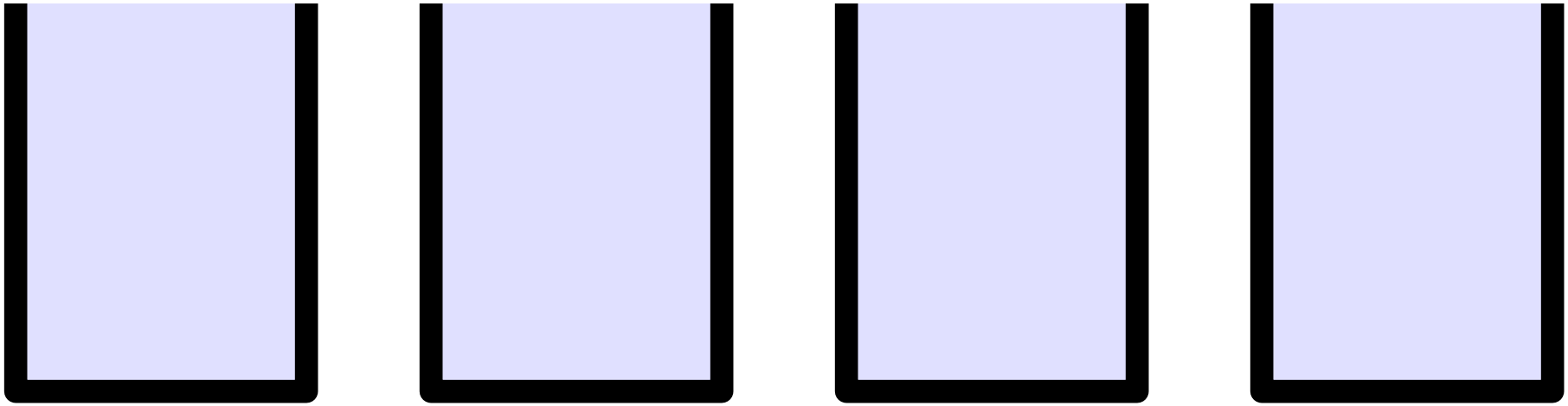
On the other hand, the expression MinCD(A) = 1+ MinCD({20,30,95}) gives 1 + 2 = 3.

Question 1

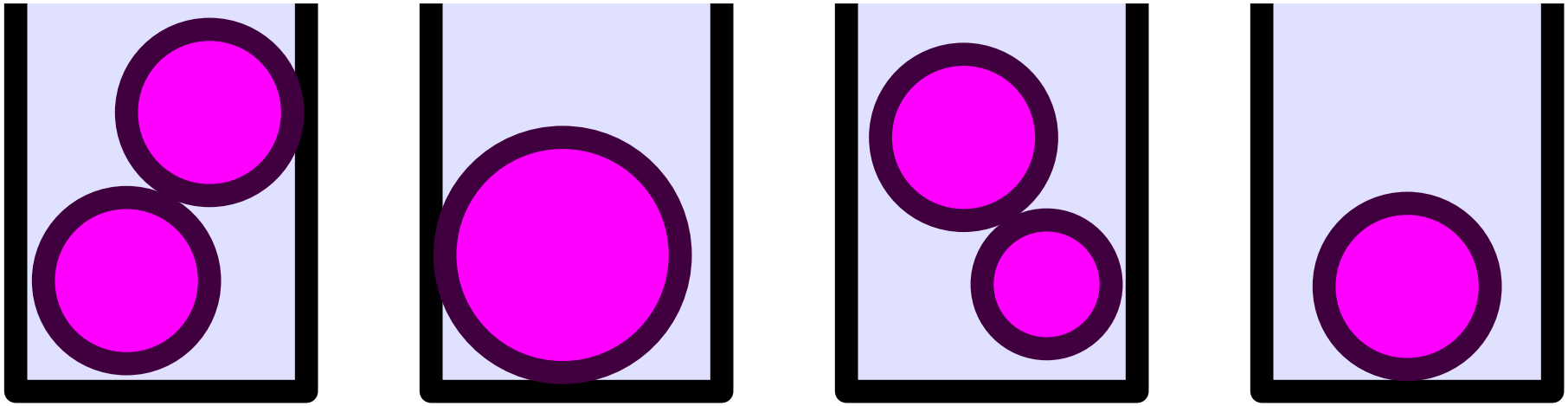
Answer: B

Answer B is correct. Consider any optimal solution.

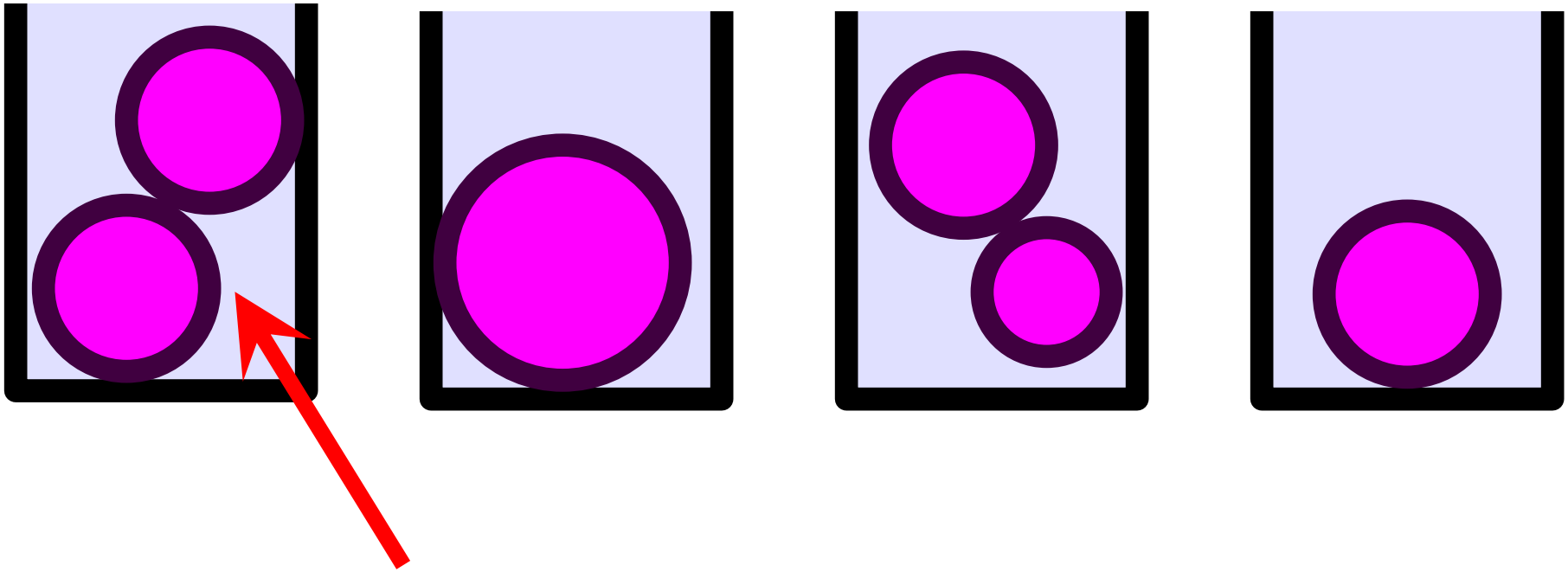
1. We can remove any pair that is in the same CD in the optimal solution.
2. The rest of the files must be stored optimally, otherwise can reduce the total number of CDs used (**optimal substructure**).



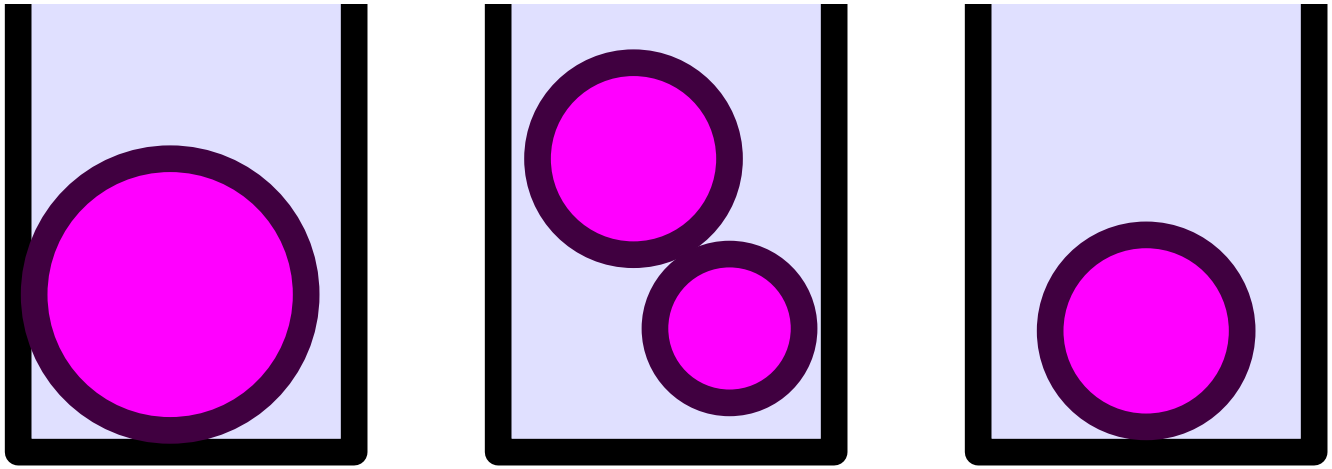
Optimal Solution:



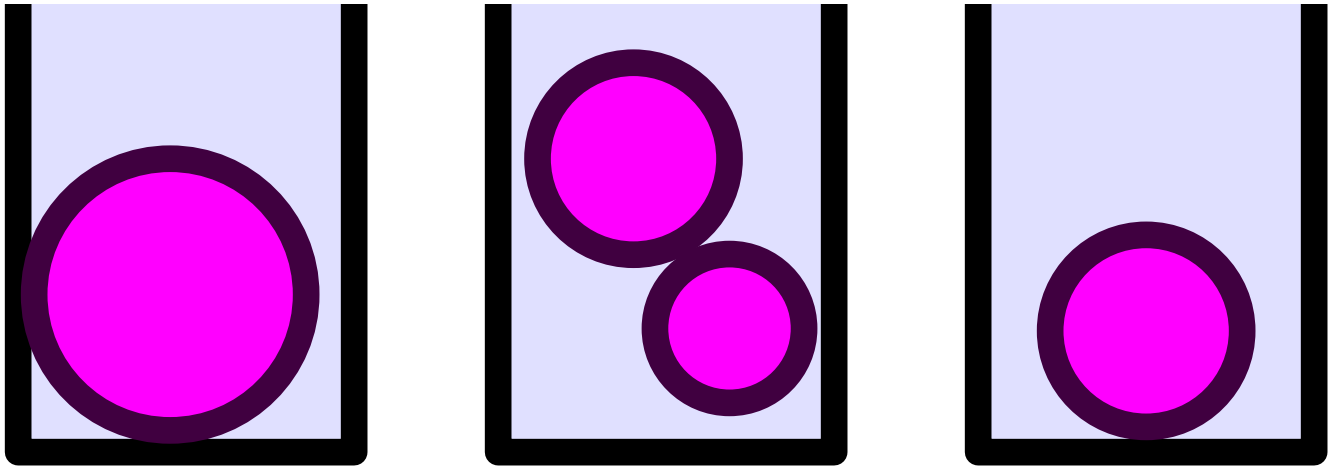
Optimal Solution:



These two are paired together
in an optimal solution

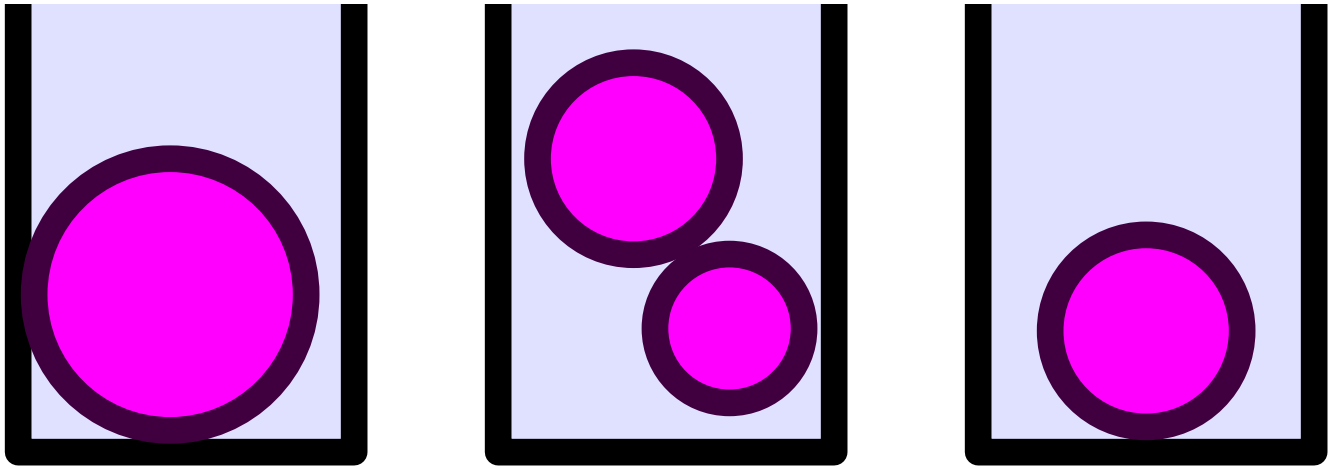


This must be optimal too!

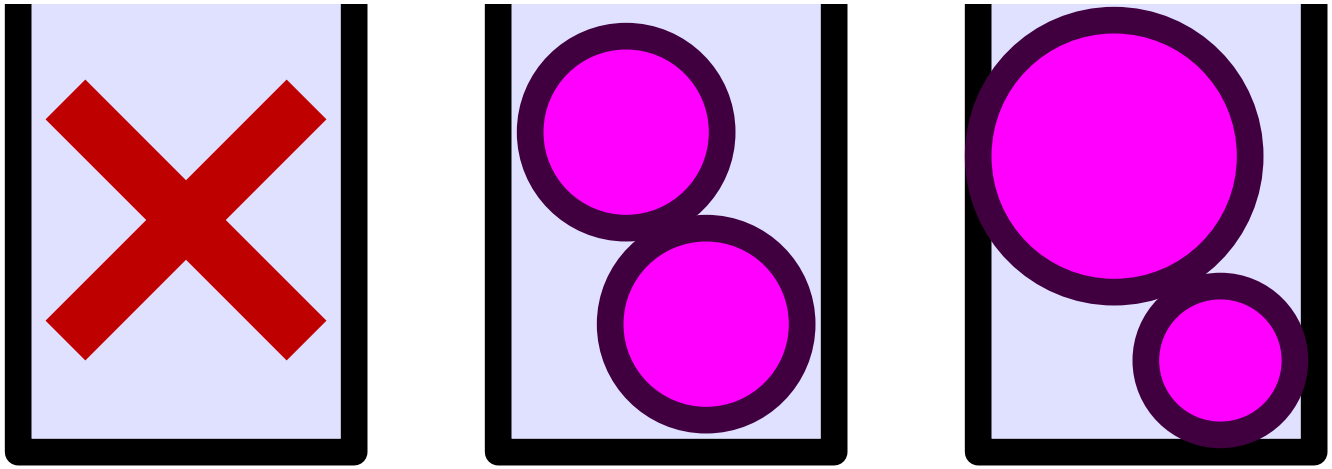


NUS
National University
of Singapore

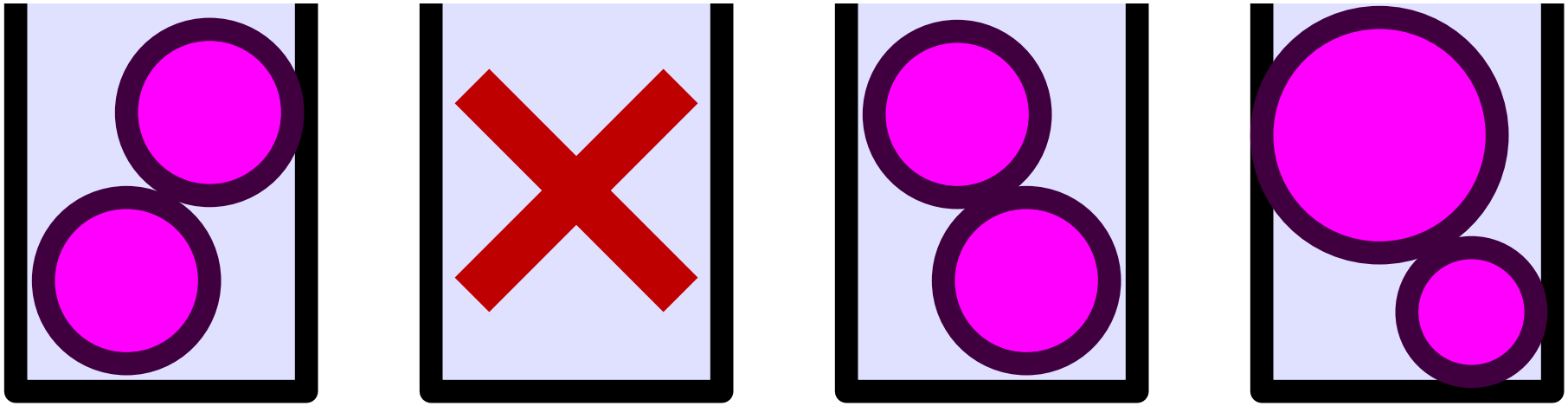
Suppose Not.



Then we can find some allocation with less CDs



Then the original is not optimal!



Question 2

Suppose Bob has a collection of music files that he wants to burn into CDs. Each CD has a storage capacity of 100MB. In addition, Bob does not want to store more than two music files per CD. Note that each music file cannot be split and hence cannot be burned on more than 1 CDs. Given a 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*.

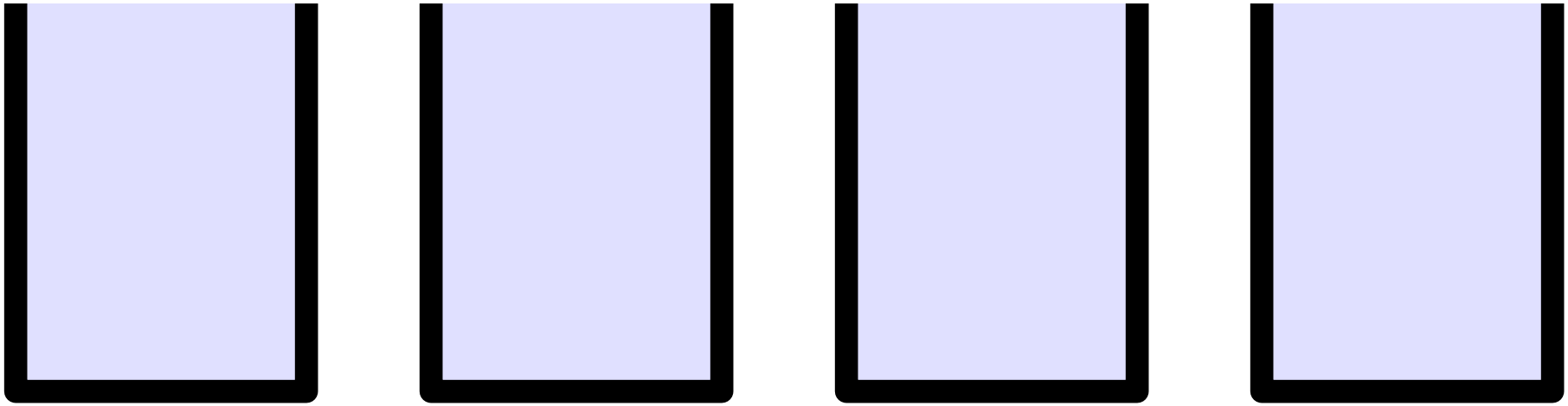
Assume that any optimal solution contains a pair that is burned onto a CD. Select all true statements about the problem.

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.

Question 2

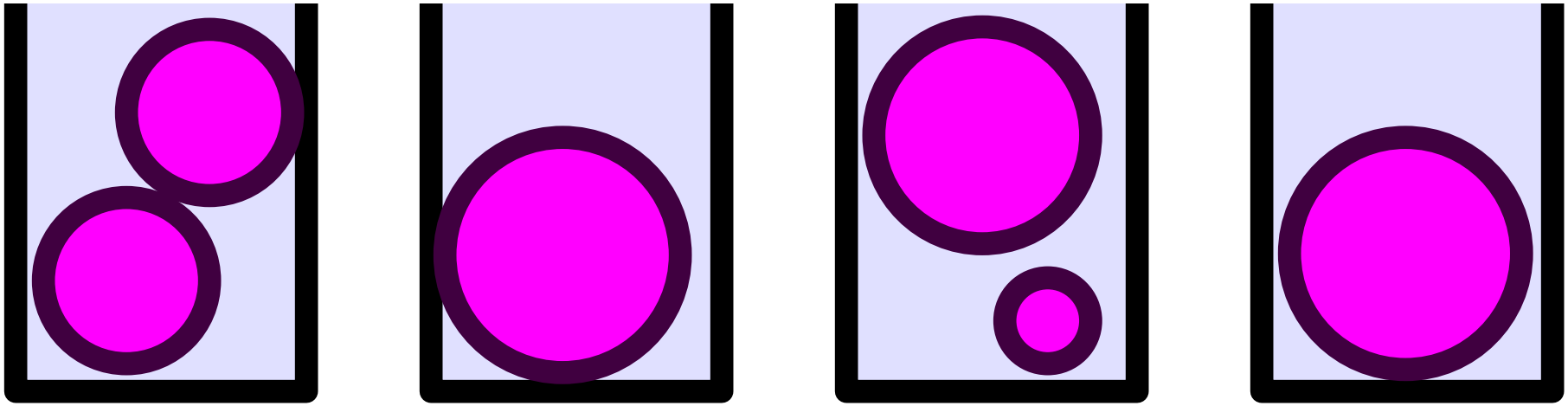
Answer: (I) and (II) only

(I) If smallest file is not included in a pair, swap it with any file that is included in a pair in an optimal solution and the number of CDs used will not change.
Hence, there exists an optimal solution that contains the smallest file in a pair, if an optimal solution with a pair exists. (**Greedy choice**)



(I)

Take any Optimal Solution

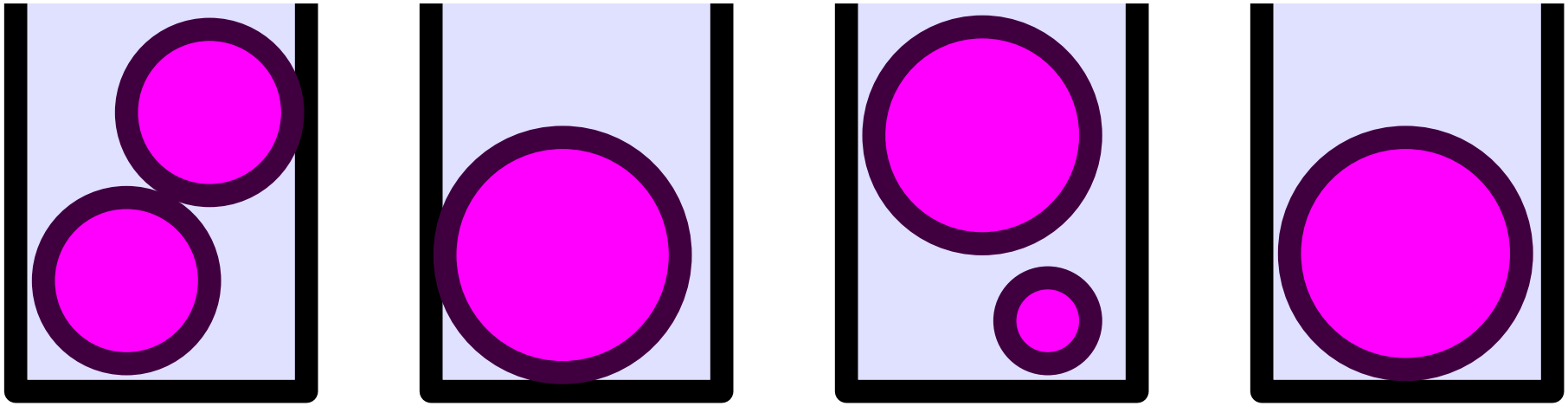


[Optimal]

(I)

Either the smallest is in a pair,

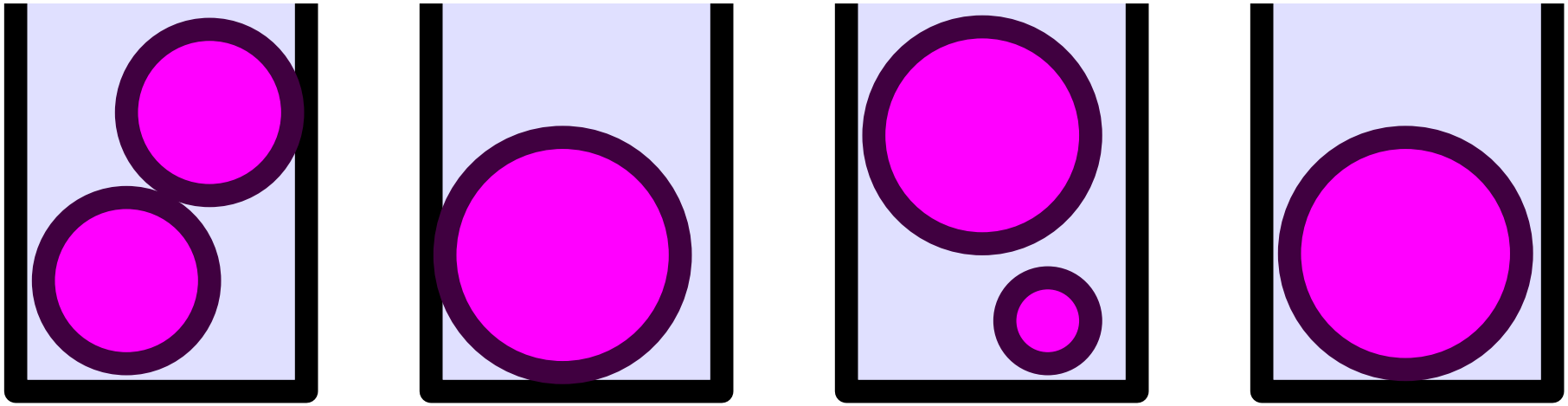
Or it is not.



[Optimal]

(I)

If it is, we are done.

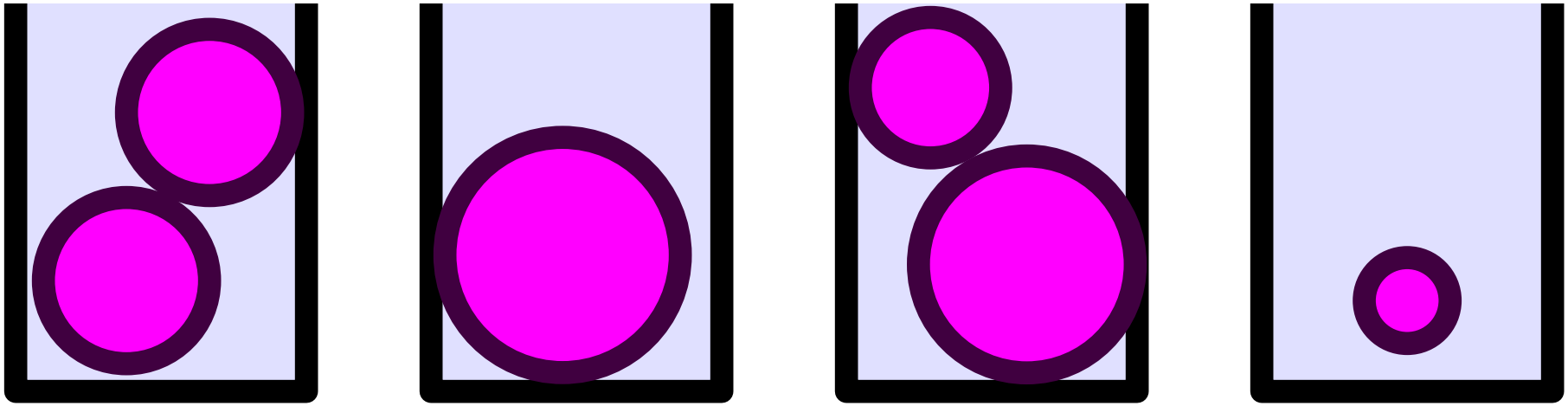


[Optimal]

(I)

NUS National University of Singapore

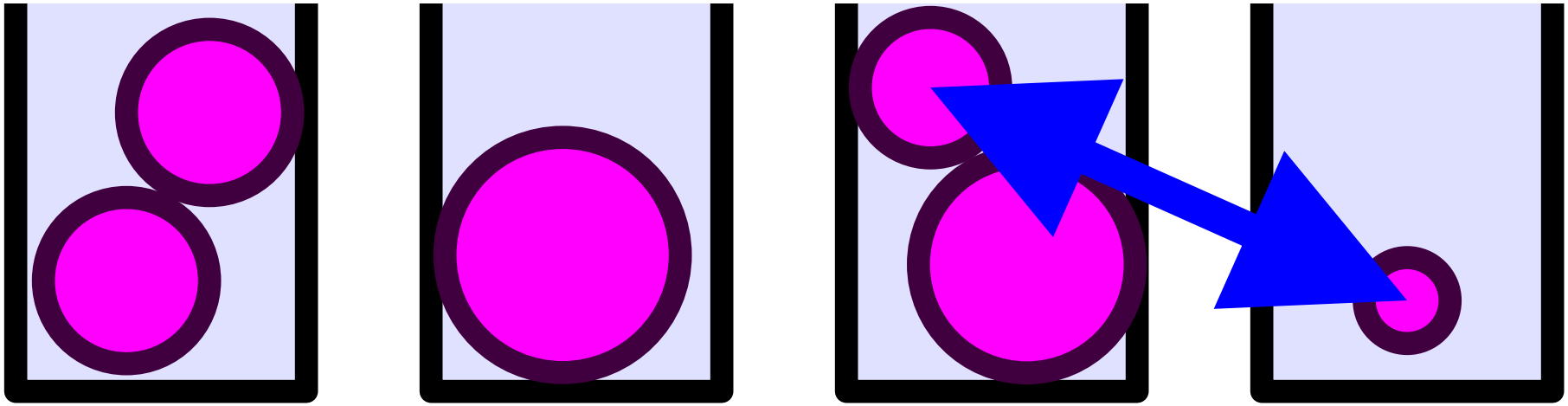
If it is not,



[Optimal]

(I)

We can always swap.



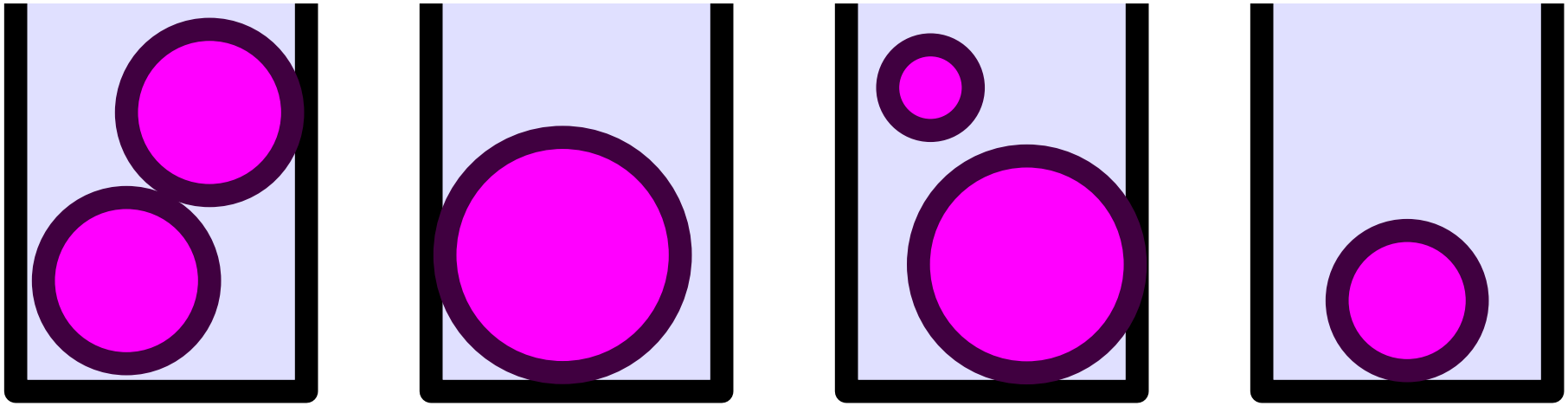
[Optimal]

(I)

This is still optimal.

There is indeed an optimal solution where

the smallest element is paired.



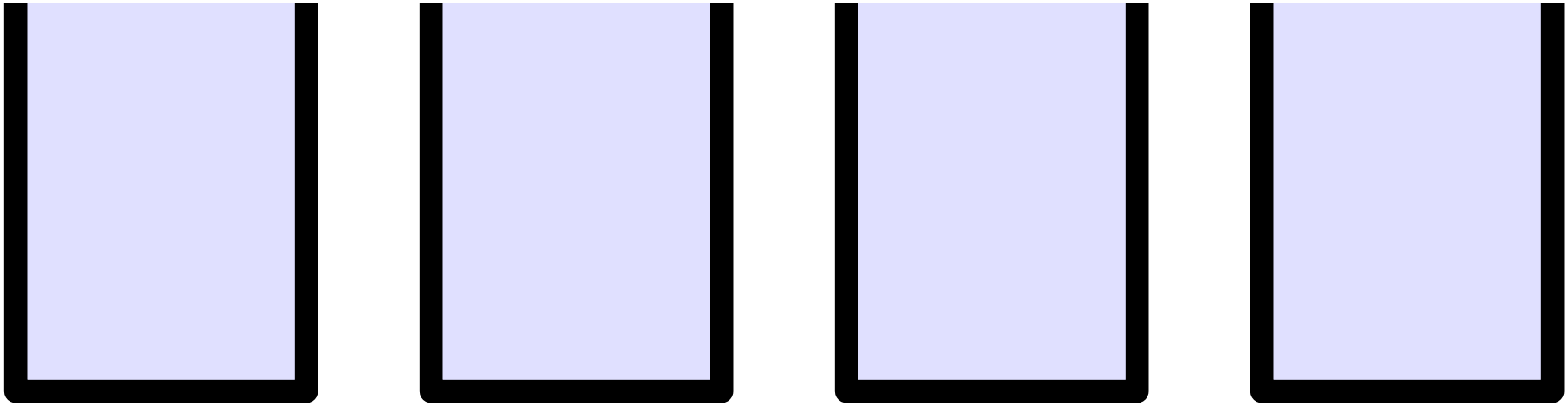
[Optimal]

Question 2

Answer: (I) and (II) only

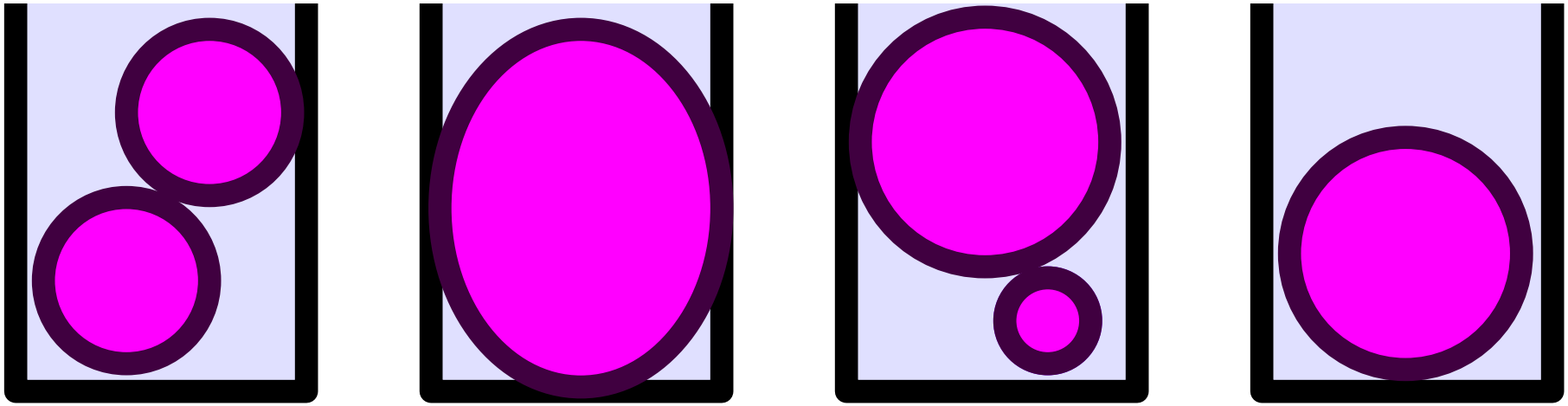
(II) From part (I), we know there is an optimal solution where smallest file is in a pair, if optimal solution contains pair.

Assume we have optimal solution where smallest file is paired with F but the largest file that fits is F'. Swap F with F'. If F' originally not paired, we still have an optimal solution. If F' paired, swap still possible as F' larger, hence F must fit in swapped CD. Hence, optimal solution with F' paired with smallest file exists. (**Greedy choice**)



(II)

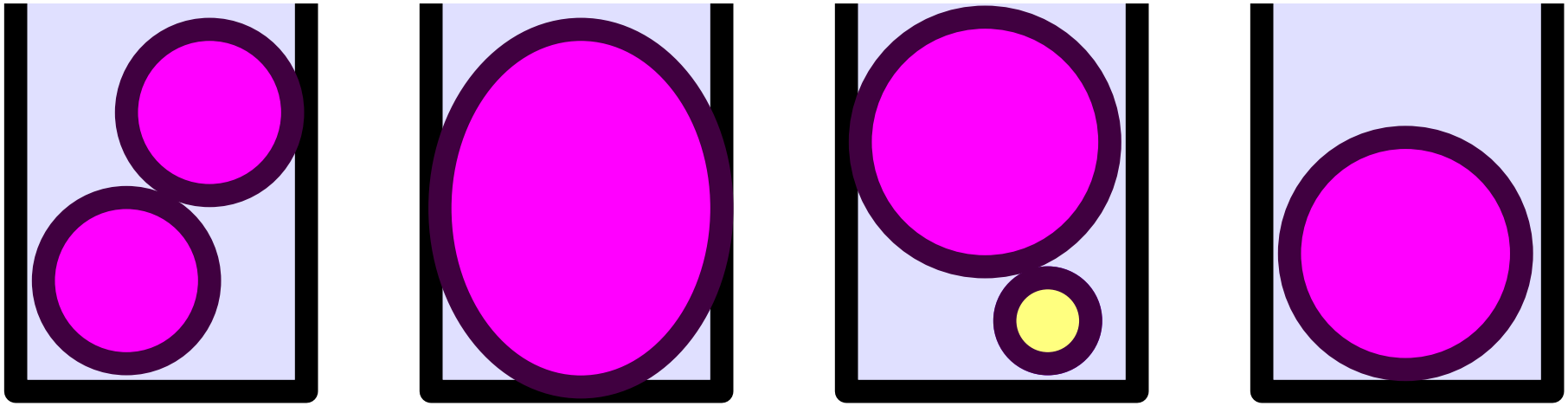
Take any Optimal Solution



[Optimal]

(II)

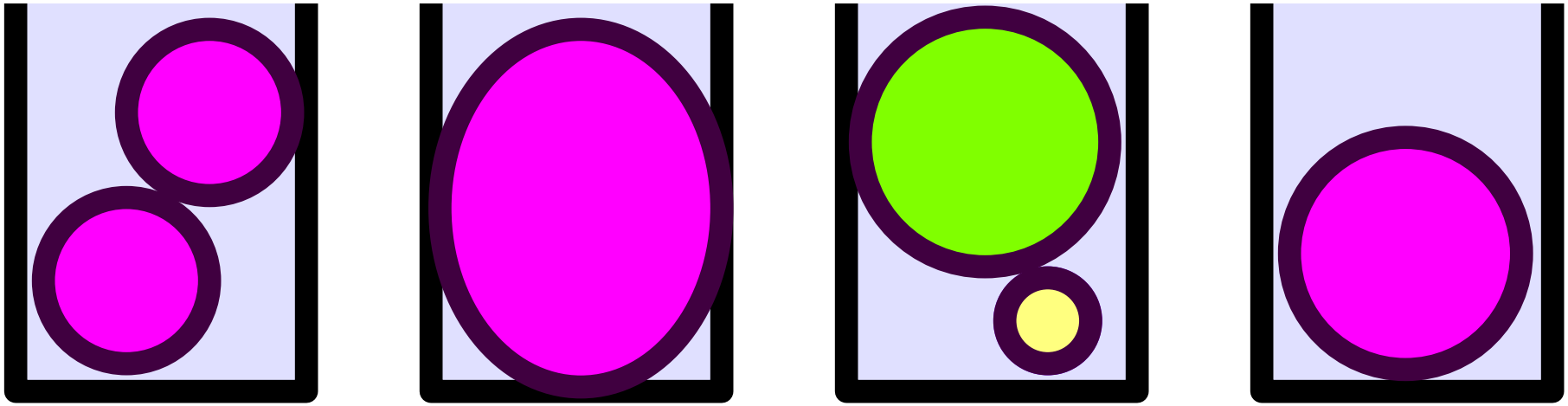
Look at the smallest element.



[Optimal]

(II)

Look at the largest element
That can fit with the smallest

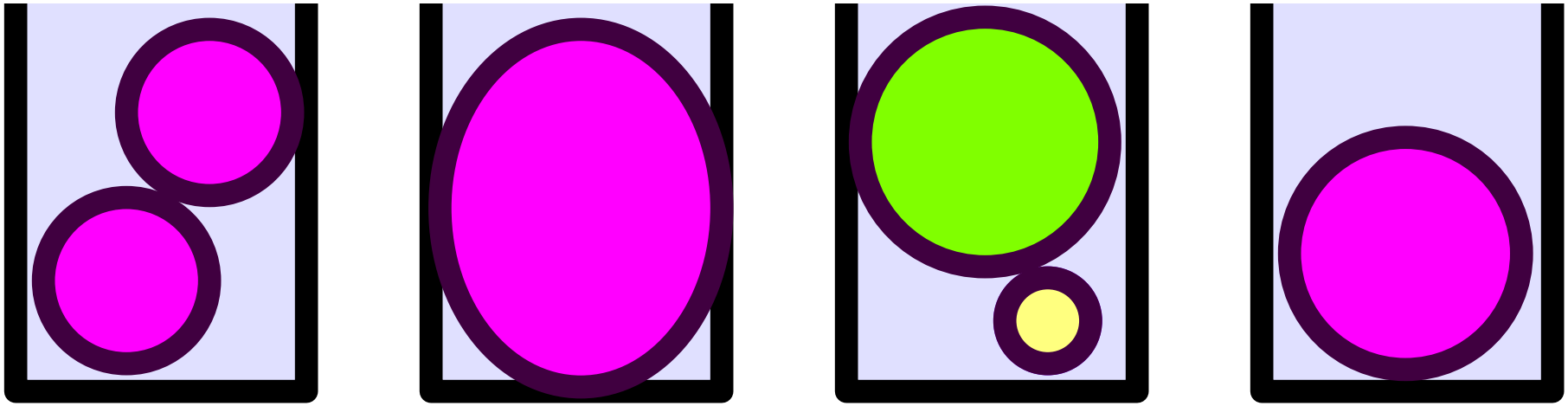


[Optimal]

(II)

They are either in the same bin,

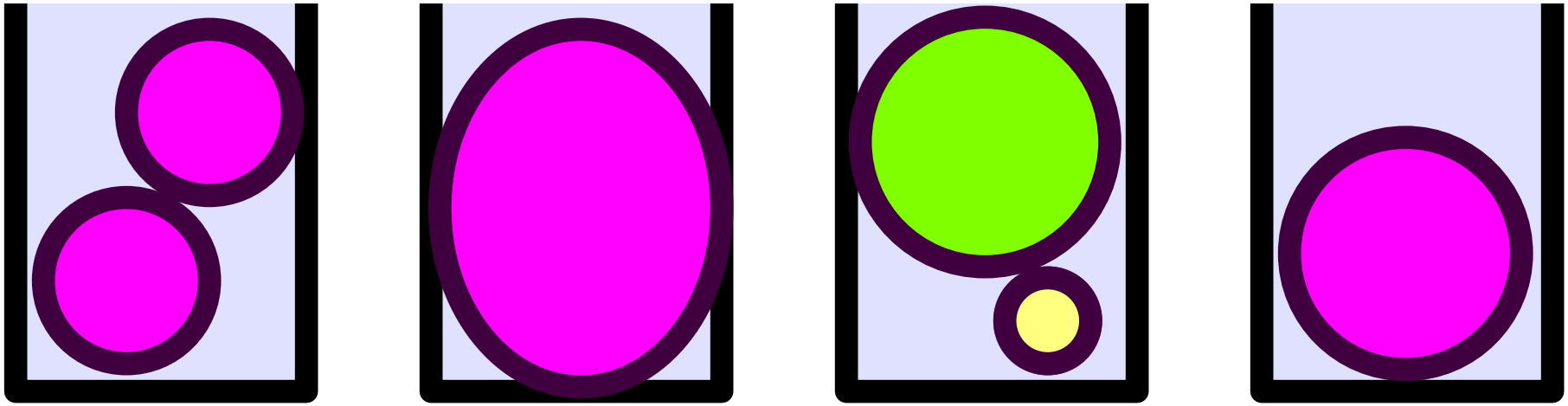
Or they are not.



[Optimal]

(II)

If they are, we are done.

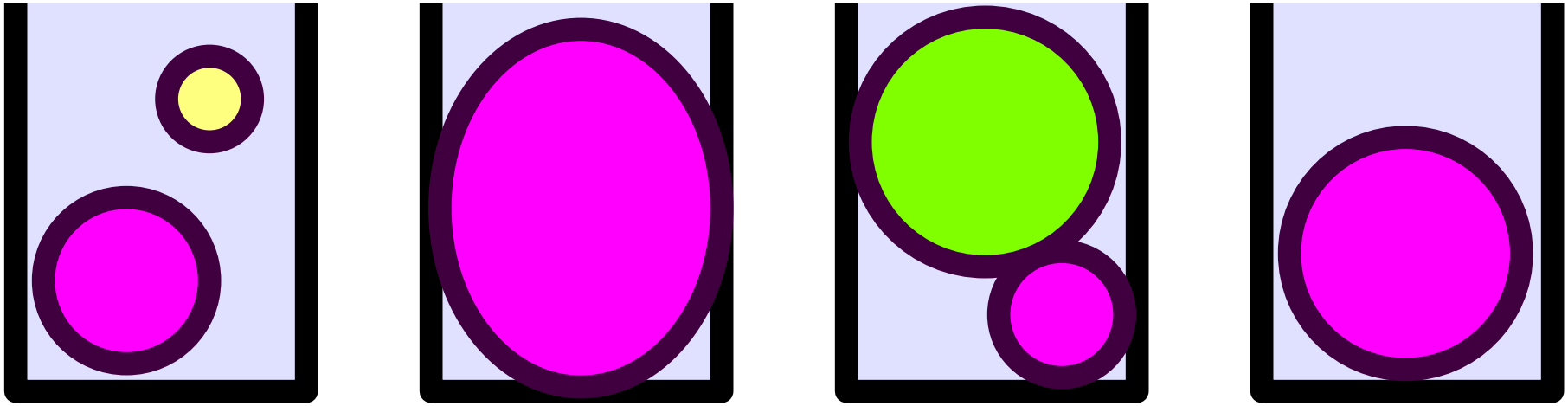


[Optimal]

(II)

NUS National University of Singapore

If they are not,

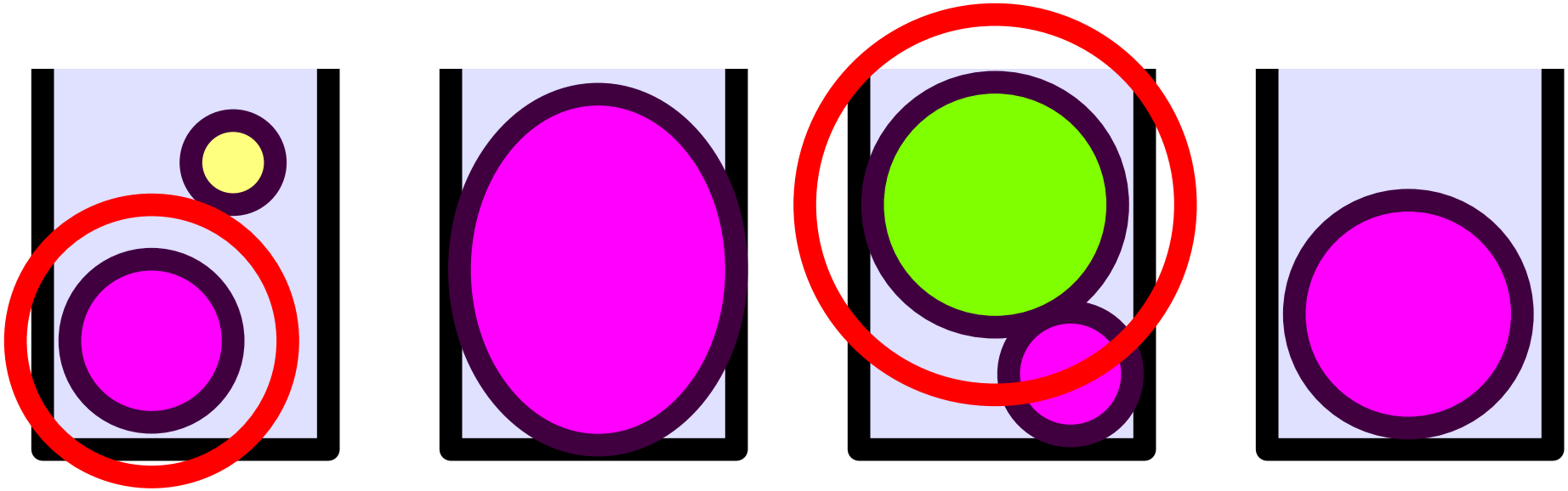


[Optimal]

(II)

NUS

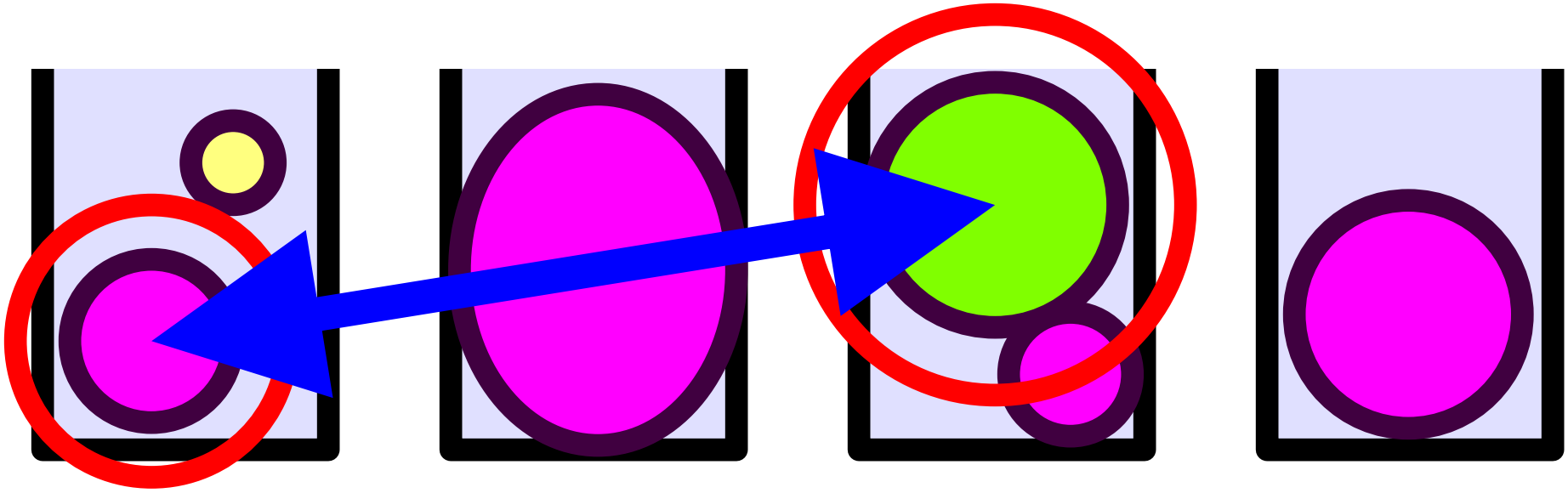
Look at these two elements.



[Optimal]

(II)

The smallest element's current partner
Can fit with the (largest element that fits with the smallest element)'s current partner.

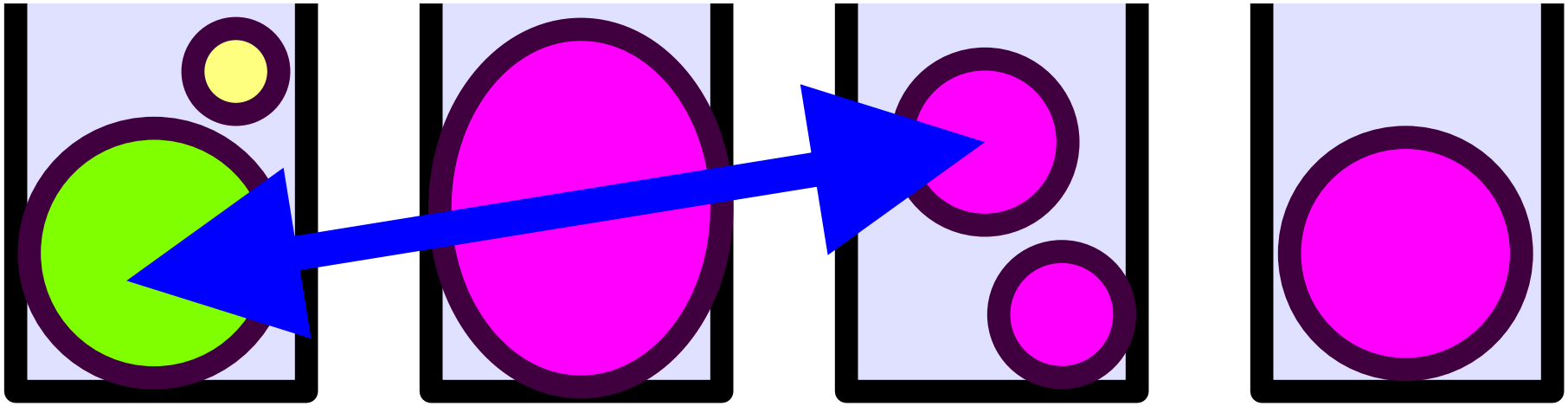


[Optimal]

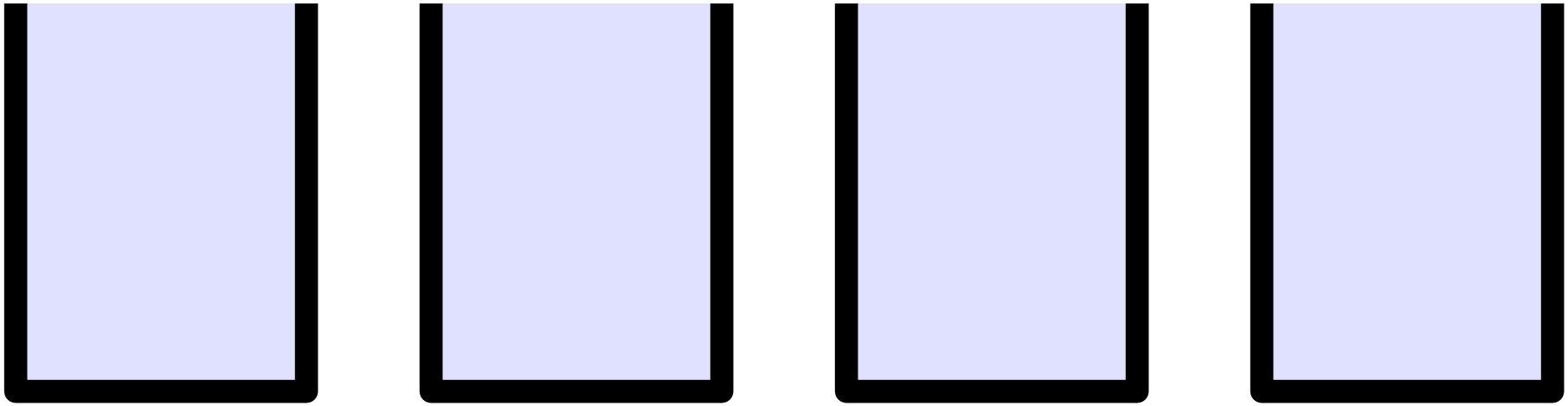
The largest element that fits with the smallest element
Can fit with the smallest element (duh)

(II)

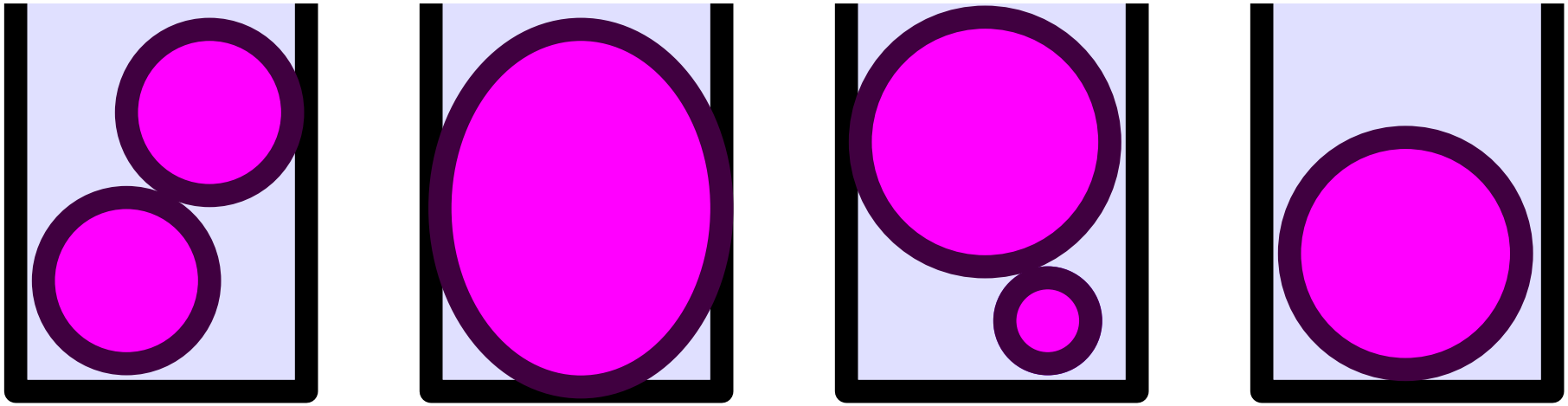
No new CDs created.
Still optimal.



[Optimal]



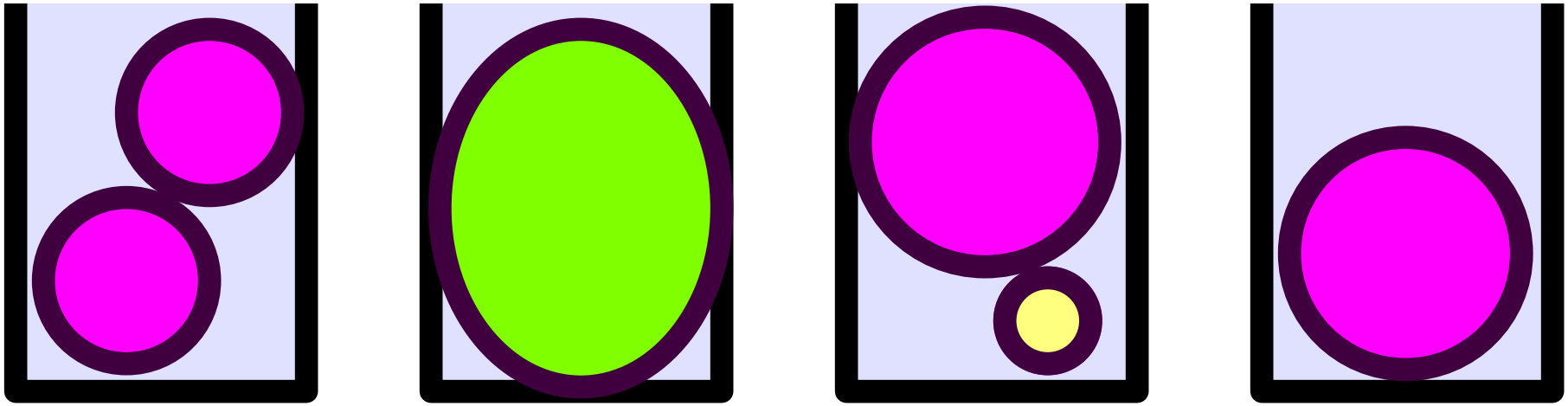
(III) NOT TRUE



(III) NOT TRUE

The smallest and largest elements

May not fit together at all!



Question 3

Suppose Bob has a collection of music files that he wants to burn into CDs. Each CD has a storage capacity of 100MB. In addition, Bob does not want to store more than two music files per CD. Note that each music file cannot be split and hence cannot be burned on more than 1 CDs. Derive the algorithm for obtaining the the minimum number of CDs that Bob needs to burn his music files on the discs and apply it to the following array of file sizes. $filesizes = [89, 59, 32, 74, 81, 12, 7, 49, 43, 51, 62, 91, 27]$ What is the minimum number of CDs required?

Question 3

Answer: A. 8

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

Sort the array

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

Use two pointers, one moving forward from start, another moving backwards from end.

Runtime $O(n \log n)$ from sorting.

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

Question 3

Answer: A. 8

[**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}

Question 3

Answer: A. 8

[**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}

Question 3

Answer: A. 8

[**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}

Question 3

Answer: A. 8

[**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}

Question 3

Answer: A. 8

[**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}

Question 3

Answer: A. 8

[**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}

Question 3

Answer: A. 8

[**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}

Question 3

Answer: A. 8

[**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}

Question 3

Answer: A. 8

[**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}

Question 3

Answer: A. 8

[**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}

Question 3

Answer: A. 8

[**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}

Question 3

Answer: A. 8

[**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}

Question 3

Answer: A. 8

[**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}**

Question 3

Answer: A. 8

[**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!**

Activity Selection Problem

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

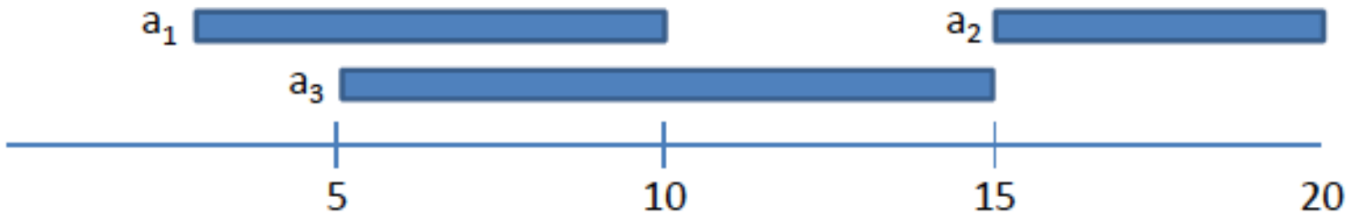
- Each activity takes place during $[s_i, f_i)$
- Two activities a_i and a_j are **compatible** if their time intervals don't overlap: $s_i \geq f_j$ or $s_j \geq f_i$.

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.

Optimal Substructure

Suppose an optimal scheduling S contains activity a_j.
Let:

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

Then, S also contains an optimal scheduling for Before_j and an optimal scheduling for After_j.

Question 4: Solution

- (1) works. Any optimal solution can be modified so that it contains the activity that starts last.

Greedy-choice property: Why?

- Consider an optimal scheduling S and let a be the activity in S that starts last.
- Replace a by a^*(the activity that starts last). The schedule remains compatible and number of activities is the same.
- The new schedule is also optimal.

Gives us a **linear time algorithm** given activities sorted in non-decreasing start time

Question 4: Solution

- (1) works. Any optimal solution can be modified so that it contains the activity that starts last.

- (2) does not work. Suppose activities are given by: [1,10), [1,5), [6,9).

- (3) does not work. Suppose activities are given by: [1,4), [3,5), [4,10).