Analysis and Design of Algorithms

Tutorial

Week 11

Question 1

In the graph coloring problem, you are given an undirected graph G = (V,E) and asked to color the vertex with a set of colors such that the colors of v_1 and v_2 must be different if $(v_1,v_2) \in E$. In the optimization version of the problem, the task is to find the minimum number of colors required to color the graph. In the decision version, the question is whether there is a coloring of the graph using *k* or fewer colors. Select all true statements.

1. If we can solve the optimization problem for graph coloring in polynomial time, we would be able to solve the decision problem for graph coloring in polynomial time.
2. If we can solve the decision problem for graph coloring in polynomial time, we would be able to solve the optimization problem for graph coloring in polynomial time.
3. If the decision problem for graph coloring cannot be solved in polynomial time, the optimization problem for graph coloring cannot be solved in polynomial time.
4. If the optimization problem for graph coloring cannot be solved in polynomial time, the decision problem for graph coloring cannot be solved in polynomial time.

Question 1

Answer: All true

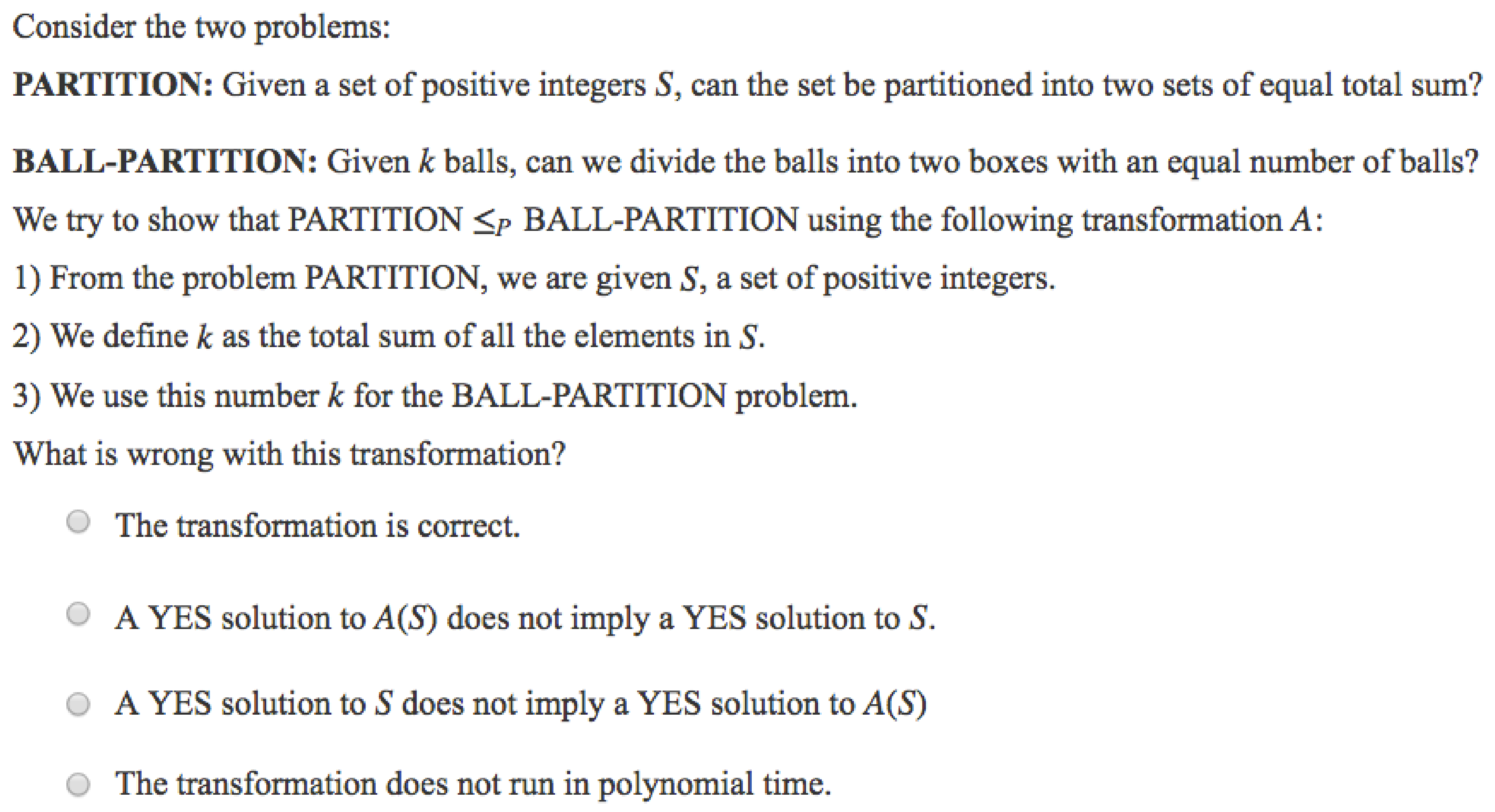
(I) Solve opt → solve decision:
Find min colors, then check whether min ≤ k

(II) Solve decision → solve opt:
Sequentially solve for number of colors to find min (or binary search)

(III) Contrapositive of (I)

(IV) Contrapositive of (II)

Question 2



Consider the two problems:

PARTITION: Given a set of positive integers S, can the set be partitioned into two sets of equal total sum?

BALL-PARTITION: Given k balls, can we divide the balls into two boxes with an equal number of balls?

We try to show that PARTITION $\leq_P$ BALL-PARTITION using the following transformation A:

1) From the problem PARTITION, we are given S, a set of positive integers.

2) We define k as the total sum of all the elements in S.

3) We use this number k for the BALL-PARTITION problem.

What is wrong with this transformation?

○ The transformation is correct.

○ A YES solution to $A(S)$ does not imply a YES solution to S.

○ A YES solution to S does not imply a YES solution to $A(S)$

○ The transformation does not run in polynomial time.

Question 2

Answer: B

Option A: Option is wrong.

The transformation is wrong. See part B

Question 2

Answer: B

Option B: Option is correct.

A YES solution to A(S) does not imply a YES solution to S.

A(S): BALL-PARTITION. **S:** PARTITION

S = {1,3}. Converted to A(S) = 4

{1,3} cannot be Partitioned,

But 4 balls can be “Ball-Partitioned” (into 2,2)

Question 2

Answer: B

Option C: Option is wrong.

If the answer to the PARTITION is YES, we can set the number of balls in each box in BALL-PARTITION to the sum in each set to get equal number of balls in each box.

Question 2

Answer: B

Option D: Option is wrong.

Just need to sum the numbers in the set for the transformation, so it’s poly time.

Question 3

PARTITION problem: Given a set *T* of nonnegative integers, can we partition *T* into two sets of equal total sum?

KNAPSACK problem: Given *n* items described by non-negative integer pairs (w_1,v_1), ... (w_n,v_n), capacity *W* and threshold *V*, is there a subset of item with total weight at most *W* and total value at least *V*?

PARTITION instances with total weights that are odd cannot be partitioned into two equal weight sets, hence can immediately be answered with a NO answer. Consider the following transformation of PARTITION instances with total weights that are even numbers into instances of KNAPSACK:

Given a PARTITION instance $\{w_1, .., w_n\}$ with total sum $S = \sum_{i=1}^{n} w_i$, construct a KNAPSACK instance (w_1,w_1), ... (w_n,w_n) with capacity $W = S/2$ and threshold $V = S/2$.

Select all true statements.

1. A YES answer to the PARTITION instance implies a YES answer to the KNAPSACK instance.
2. A YES answer to the KNAPSACK instance implies a YES answer to the PARTITION instance
3. The transformation runs in polynomial time

Question 3

Answer: (I), (II), (III) only

- **(I)** YES answer to PARTITION implies YES answer to KNAPSACK:

Simply use the same subset in PARTITION for knapsack -- will get weight S/2 and value S/2

Question 3

Answer: (I), (II), (III) only

- **(II)** YES answer to KNAPSACK implies YES answer to PARTITION:

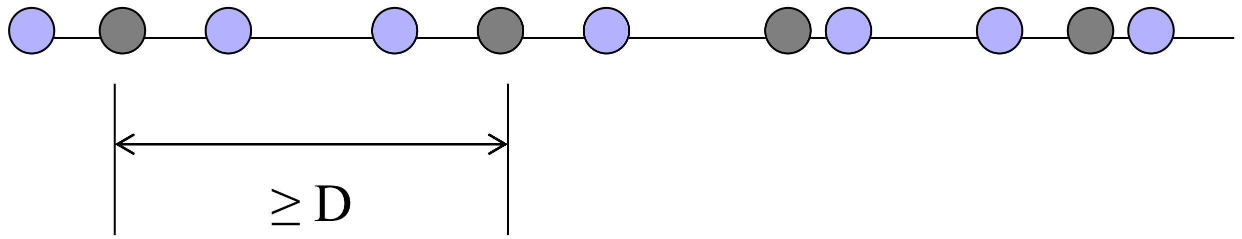
YES answer to KNAPSACK means there is a subset with weight no more than S/2 with value at least S/2. Since weight equals value in the transformed instances, only way that can happen is if weight and value equal S/2. Same subset can be used for PARTITION.

Question 3

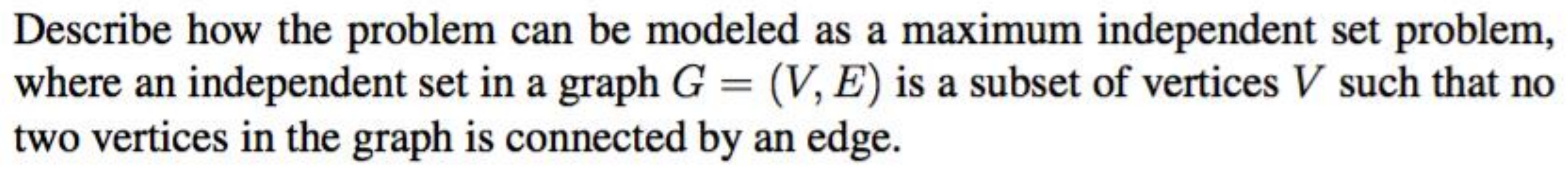
- **(III):** Transformation simply copies weight to value, so poly time.

Question 4

YuckDonald wants to open as many of its chain restaurant on Orchard Road as possible. It has found n suitable locations for its restaurants $a_1, \ldots, a_n$. The restaurants should be at least D distance apart so that they do not compete with each other.

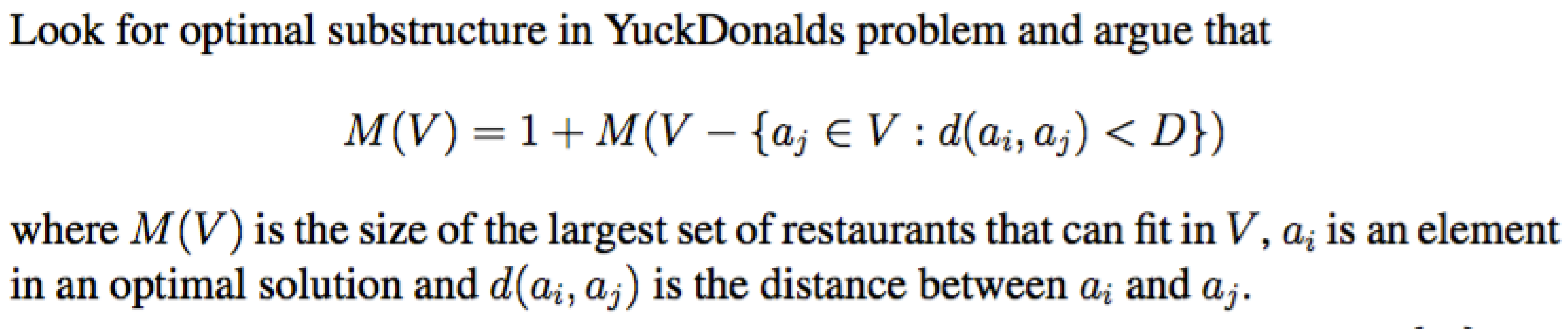


Question 4(a)


Describe how the problem can be modeled as a maximum independent set problem, where an independent set in a graph $G = (V, E)$ is a subset of vertices V such that no two vertices in the graph is connected by an edge.

Solution: Let each vertex v_i in V represent a location a_i. Two vertices v_i and v_j are connected by an edge if the distance between a_i and a_j is less than D. Finding the largest independent set in this graph corresponds to finding the largest number of locations with distance at least d apart.

Question 4(b)

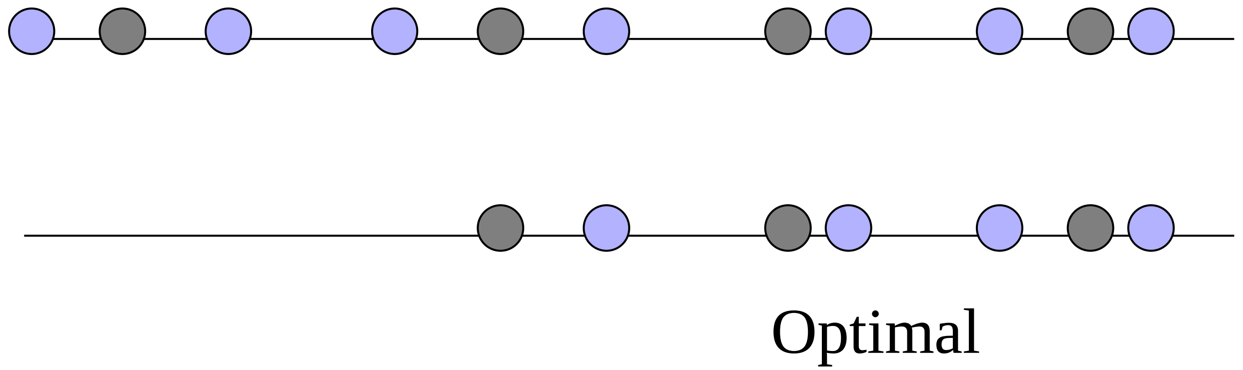


Look for optimal substructure in YuckDonalds problem and argue that

$$M(V) = 1 + M(V - \{a_j \in V : d(a_i, a_j) < D\})$$

where $M(V)$ is the size of the largest set of restaurants that can fit in V, a_i is an element in an optimal solution and $d(a_i, a_j)$ is the distance between a_i and a_j.

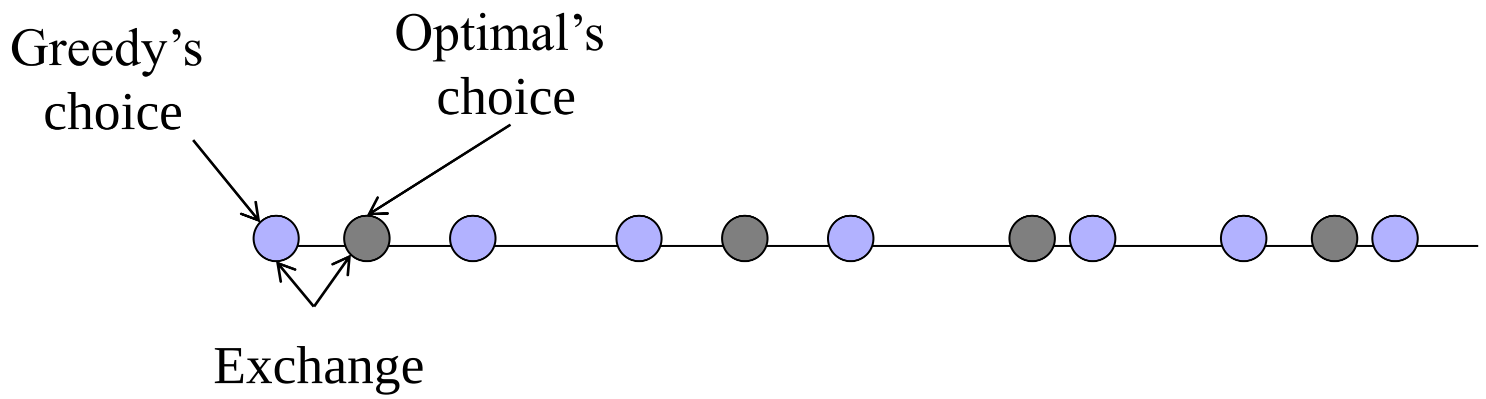
Solution: Consider an optimal solution, which is an independent set I. If we remove a location a_i from I, $I - \{a_i\}$ is a independent set for the set $V - \{a_j : d(a_i, a_j) < D\}$ and is the optimal solution. Otherwise, we can improve the solution $I - \{a_i\}$ and create a larger solution to the original problem by adding a_i to it.



Question 4(c)

Assume that the locations for a set of restaurant $a_1, \ldots, a_n$ is sorted and lie on a straight line. Show that a_1 is part of some optimal solution, i.e. selecting the smallest element satisfies the greedy choice property.

Solution: Consider an optimal solution. If a_1 is in the solution, we are done. If not, find the smallest value a_j that is in the optimal solution, and swap it with a_1. We will obtain another feasible solution as a_1 would be further from other elements in the solution compared to a_j. As the number of elements is unchanged, this solution is also optimal.



Question 4(d)

Complete your greedy algorithm for evaluating $M(V)$ and argue its correctness by mathematical induction.

Algorithm

1. Add the smallest element
2. Remove all elements with distance less than D from removed element.
3. Repeat until no element left

Correctness

1. Base case: Empty set. No element needed.
2. Inductive hypothesis: Greedy algorithm correct when $|V| < k$.

3. Consider set V with $|V| = k$.
   - Greedy choice property implies that smallest element is part of an optimal independent set.
   - Optimal substructure implies that after removing smallest element and elements of distance less than D to it, we need to solve for the remaining set optimally.
   - Inductive hypothesis implies we solve for the remaining set optimally.

Question 4(e)

We get a poly-time greedy algorithm for this shop opening problem and as a consequence we get a poly-time algorithm for the maximum independent set problem. Is this true?

1. Yes
2. No

Question 4(e)

Answer: (2) NO

Since Shop Opening problem is poly-time reducible to the maximum independent set problem, not the converse.