Solution set

Competitive Programming: Core Skills

Artur Riazanov

SPbSU

Introduction

- Combinatorial problems are problems where only finite objects involved.

Introduction

- Combinatorial problems are problems where only finite objects involved.
- We will learn how to figure out the set of objects involved in the problem and enumerate them all.

Introduction

- Combinatorial problems are problems where only finite objects involved.
- We will learn how to figure out the set of objects involved in the problem and enumerate them all.
- On the contrary, geomery problems are about infinite objects and our approach will not work for them.

Introduction

- Combinatorial problems are problems where only finite objects involved.
- We will learn how to figure out the set of objects involved in the problem and enumerate them all.
- On the contrary, geomery problems are about infinite objects and our approach will not work for them.
- The approach is pretty slow but it's conceptually correct by definition.

Search space

1. Find an element of a set A such that is satisfies some property.

Search space

1. Find an element of a set A such that is satisfies some property.
2. Find an element of a set A such that some function is minimized/maximized.

Search space

1. Find an element of a set A such that is satisfies some property.
2. Find an element of a set A such that some function is minimized/maximized.
3. Find the number of elements of a set A satisfying some property.

Search space

1. Find an element of a set A such that is satisfies some property.
2. Find an element of a set A such that some function is minimized/maximized.
3. Find the number of elements of a set A satisfying some property.

We will call the set A search space.

Examples

Let's look at some examples of problems.

Examples

Let's look at some examples of problems.

Superstring

Given m strings $s_1, \ldots, s_m$ consisting of letters "a" and "b" only and an integer n. Find a string s of length n containing each s_i (for all $1 \leq i \leq m$) as a substring.

Examples

Let's look at some examples of problems.

Superstring

Given m strings $s_1, \ldots, s_m$ consisting of letters "a" and "b" only and an integer n. Find a string s of length n containing each s_i (for all $1 \le i \le m$) as a substring.

Input: $m = 2$; $n = 3$; $s_1 =$ `ab`, $s_2 =$ `ba`

Output: There are two possible answers: `aba` (`aba`, `aba`) and `bab` (`bab`, `bab`).

Examples

Segment with the largest sum

Given an array $a_1, \ldots, a_n$. What is the largest possible sum $a_l + a_{l+1} + \ldots + a_{r-1} + a_r$ for $1 \le l \le r \le n$? Note that a_i could be negative.

Input: $a = (4, 1, -2, 3, -10, 5)$
Ouput: The best segment is $(\underbrace{4, 1, -2, 3}, -10, 5)$ and the sum is $4 + 1 + (-2) + 3 = 6$.

Examples

Robber's problem

you have a knapsack of volume V and n items of volumes $v_1, \ldots, v_n$ and costs $c_1, \ldots, c_n$. What is the largest possible cost of the set of items, which fits into the knapsack?

Examples

Robber's problem

you have a knapsack of volume V and n items of volumes $v_1, \ldots, v_n$ and costs $c_1, \ldots, c_n$. What is the largest possible cost of the set of items, which fits into the knapsack?

Input: $V = 5$; $n = 3$

$v_1 = 3 \quad v_2 = 2 \quad v_3 = 5$

$c_1 = 2 \quad c_2 = 3 \quad c_3 = 6$

Output: The best solution is to put the last item to the knapsack and get the total cost 6.

Search space

- One way to solve a problem is to simply go through all possible candidate solutions. For the superstring problem, the search space consists of all strings of length n over the alphabet $\{a, b\}$. For each such string, we check whether it is indeed a superstring of $s_1, \ldots, s_m$

Search space

- One way to solve a problem is to simply go through all possible candidate solutions. For the superstring problem, the search space consists of all strings of length n over the alphabet $\{a, b\}$. For each such string, we check whether it is indeed a superstring of $s_1, \ldots, s_m$
- Let's start with the first problem with $n = 4$, $s_1 =$ `bab`, $s_2 =$ `abb`.

Search space

- One way to solve a problem is to simply go through all possible candidate solutions. For the superstring problem, the search space consists of all strings of length n over the alphabet $\{a, b\}$. For each such string, we check whether it is indeed a superstring of $s_1, \ldots, s_m$
- Let's start with the first problem with $n = 4$, $s_1 = \texttt{bab}$, $s_2 = \texttt{abb}$.
- There are only $2^4 = 16$ strings of four letters "a" and "b".

Search space

Candidate	bab	abb	Candidate	bab	abb
aaaa	×	×	baaa	×	×
aaab	×	×	baab	×	×
aaba	×	×	baba	baba	×
aabb	×	aabb	**babb**	babb	babb
abaa	×	×	bbaa	×	×
abab	abab	×	bbab	bbab	×
abba	×	×	abba	×	×
abbb	×	abbb	bbbb	×	×

Search space

- So we've figured out that the answer for the superstring problem is `babb` by checking all candidates solutions.

Search space

- So we've figured out that the answer for the superstring problem is `babb` by checking all candidates solutions.
- The rest two problems are in the second category: we have to find an objects maximizing given function.

Search space

- So we've figured out that the answer for the superstring problem is `babb` by checking all candidates solutions.
- The rest two problems are in the second category: we have to find an objects maximizing given function.
- The search space is not given as explicitly as in the superstring problem.

Search space

Problem	Search space
Superstring	strings containing letters “a” and “b”

Search space

Problem	Search space
Superstring	strings containing letters “a” and “b”
Robber’s problem	all possible sets of items

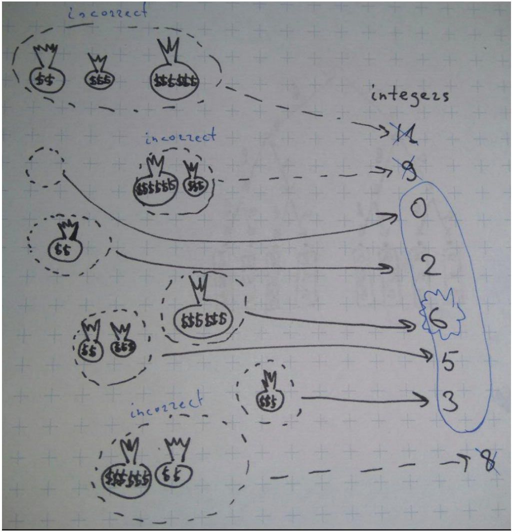
Search space

Problem	Search space
Superstring	strings containing letters “a” and “b”
Robber's problem	all possible sets of items
Segment with the largest sum	pairs (l, r) such that $l \leq r$

Search space

Problem	Search space
Superstring	strings containing letters “a” and “b”
Robber’s problem	all possible sets of items
Segment with the largest sum	pairs (l, r) such that $l \leq r$

Robber's problem



What is search space for?

- Now we can easily find the segment with the largest sum. We will just try all possible pairs l, r, calculate the sum $a_l + a_{l+1} + \ldots + a_r$ for each of them and choose the largest value.

What is search space for?

- Now we can easily find the segment with the largest sum. We will just try all possible pairs l, r, calculate the sum $a_l + a_{l+1} + \ldots + a_r$ for each of them and choose the largest value.
- We can try all possible pairs with two embedded `for` cycles.

What is search space for?

- Now we can easily find the segment with the largest sum. We will just try all possible pairs l, r, calculate the sum $a_l + a_{l+1} + \ldots + a_r$ for each of them and choose the largest value.
- We can try all possible pairs with two embedded `for` cycles.
- For the substring problem we want to try all possible strings of n symbols.

What is search space for?

- Now we can easily find the segment with the largest sum. We will just try all possible pairs l, r, calculate the sum $a_l + a_{l+1} + \ldots + a_r$ for each of them and choose the largest value.
- We can try all possible pairs with two embedded `for` cycles.
- For the substring problem we want to try all possible strings of n symbols.
- It'd be good to have n embedded `for` cycles iterating from letter "a" to letter "b".

n embedded for cycles

Your pseudo-Python code will look like this for the superstring problem:

```
for x[0] in ['a','b']:
    for x[1] in ['a','b']:
        # ...
        for x[n-1] in ['a','b']:
            # check if x contains
            # all strings s[1], ... s[m]
```

n embedded for cycles

Your pseudo-Python code will look like this for the superstring problem:

```
for x[0] in ['a','b']:
    for x[1] in ['a','b']:
        # ...
        for x[n-1] in ['a','b']:
            # check if x contains
            # all strings s[1], ... s[m]
```

Unfortunatelly, we can not do it just like that in real Python. But let's learn how to write things like that and finally obtain recipy how to solve combinatorial problems in the next video!