Brute force algorithm

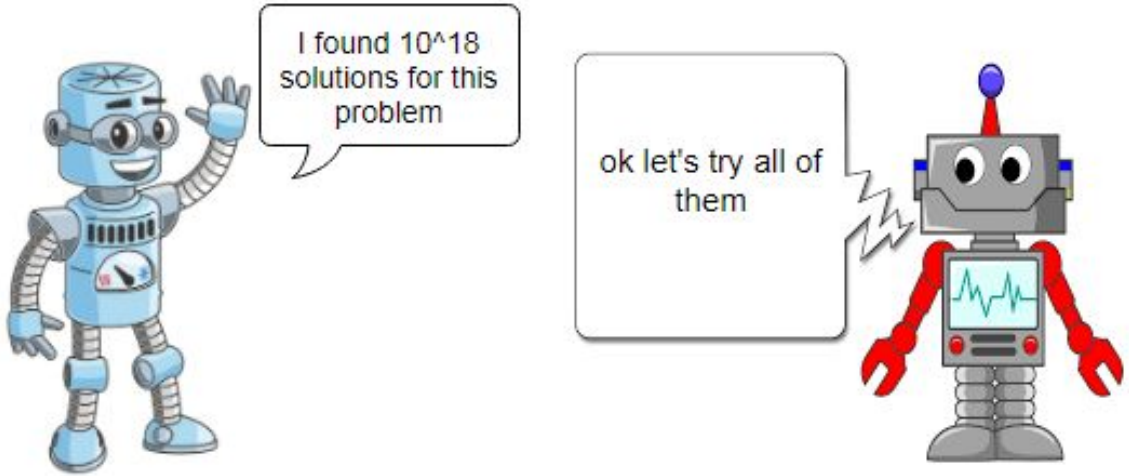
Nguyen Phuc Dat	18502573	Instructors:	Nguyen Thanh Son
Dong Quoc Tuan	18520185		Le Dinh Duy
Huynh Minh Tri	18520176	Course:	Design and Analysis of Algorithms

Table of contents

Introduction.

The “Brute-Force” algorithm is actually the most straightforward approach to solving a that rely on sheer computing power and trying every possibility rather than advanced techniques to improve efficiency.

BigO: Space complexity of the problem



Pattern recognition.

How Brute force solve problem P?

Step by step:

In order candidate for *P* after the current one *c*.

1. *valid* (*P*, *c*): check whether candidate *c* is a solution for *P*.
2. *output* (*P*, *c*): use the solution *c* of *P* as appropriate to the application.

Pseudocode

```
c ← first(P)
while c ≠ Λ do
    if valid(P,c) then
        output(P, c)
    c ← next(P, c)
end while
```

Pros and Cons

➔ Straightforward: easy to implement.

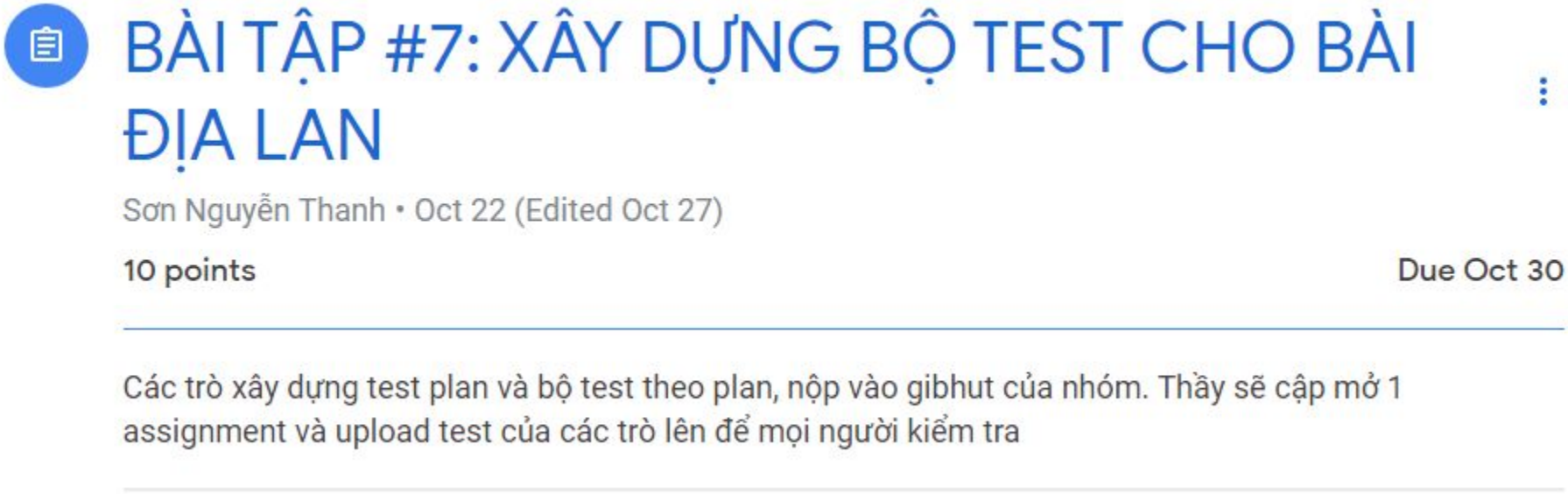
➔ For some important problems, with no limitation on instance size, brute force could achieve a result with high accuracy.

➔ Take too much time to execute

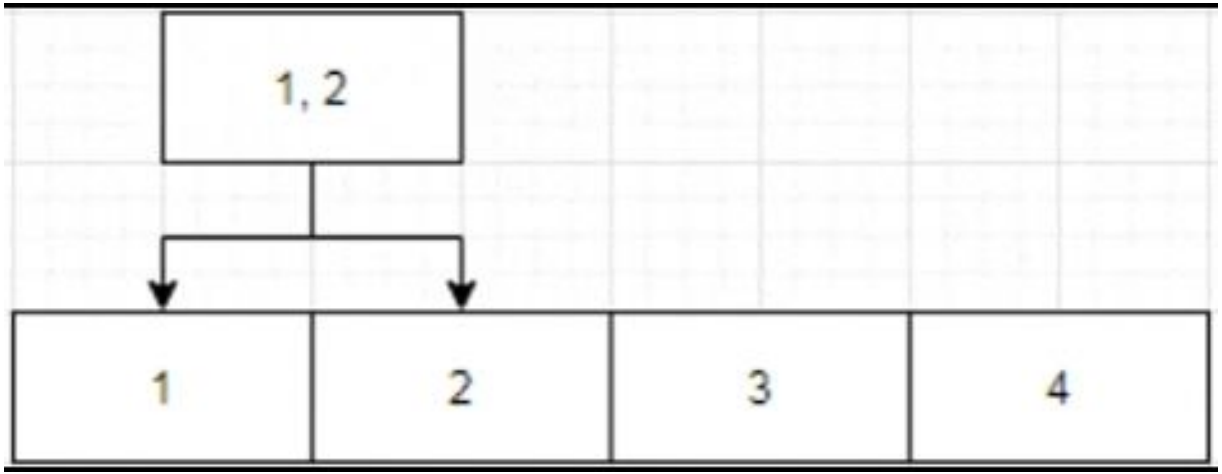
➔ Brute force can not applied to **most** real-world problems. (Ex: In real-life, we **can not** try all solutions to find the suitable answer for our problem).

Suitable Case

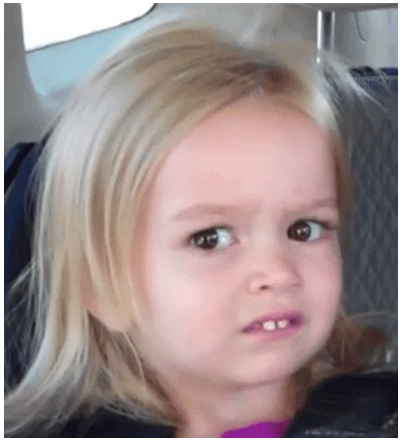
Check the correction of the program by test suite.



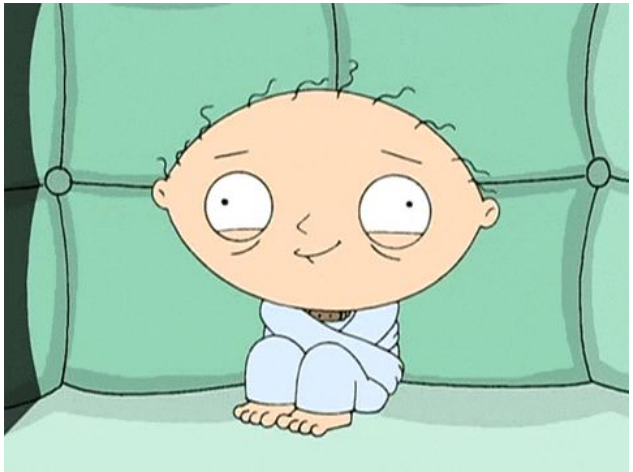
Constraints: n = 4, k = 2, arr = [1, 2, 3, 4]



```
START
INPUT arr, k
Create array combs contains all combinations of
size k from arr
FOR EACH c in combs:
     IF (call check_result(c)) equal TRUE:
          PRINT YES
     ELSE:
          PRINT NO
END
```



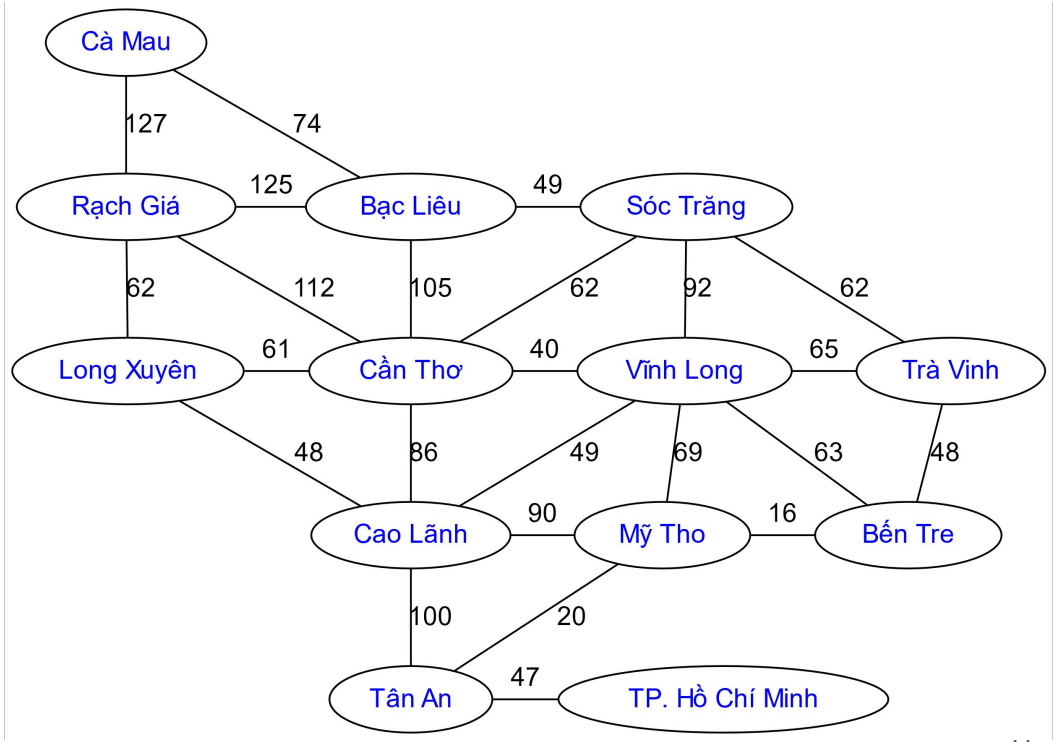
```
FUNCTION check_result(k_items):
    Create variable res = k_items[0]
    FOR each item in k_items:
        Assign res = res and item
    IF res equal 0:
        return TRUE
    ELSE:
         return FALSE
```



Unsuitable Case

Finding a shortest path from Cà Mau to TP.Hồ Chí Minh

- Brute force: O(E^V)
- Dijkstra's algorithm: O(V^2)
- A* search: O(E)



Applications

Crack password zip File

```
Create variable c contains all character on keyboard.

c = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890!@#$%^&()'

Input minLength, maxLength

For l from minLength to maxLength:

    tryPass = Cartesian product of c with itself for l times

    if tryPass is real password:

        Return tryPass
```

source code: https://github.com/mnismt/CompressedCrack

Hyperparameter Tuning

Libraries: GridSearchCV(scikit-learn), ...

Initialize space for parameter:

Discretize domain of parameter

C ∈ 0.1...0.5 => [0.1, 0.2, 0.3, 0.4, 0.5]

Alpha ∈ 0.1...0.4 => [0.1, 0.2, 0.3, 0.4]

C \ Alpha	0.1	0.2	0.3	0.4
0.5	0.701	0.703	0.697	0.696
0.4	0.699	0.702	0.698	0.702
0.3	0.721	0.726	0.713	0.703
0.2	0.706	0.705	0.704	0.701
0.1	0.698	0.692	0.688	0.675

Brute force in creating test case

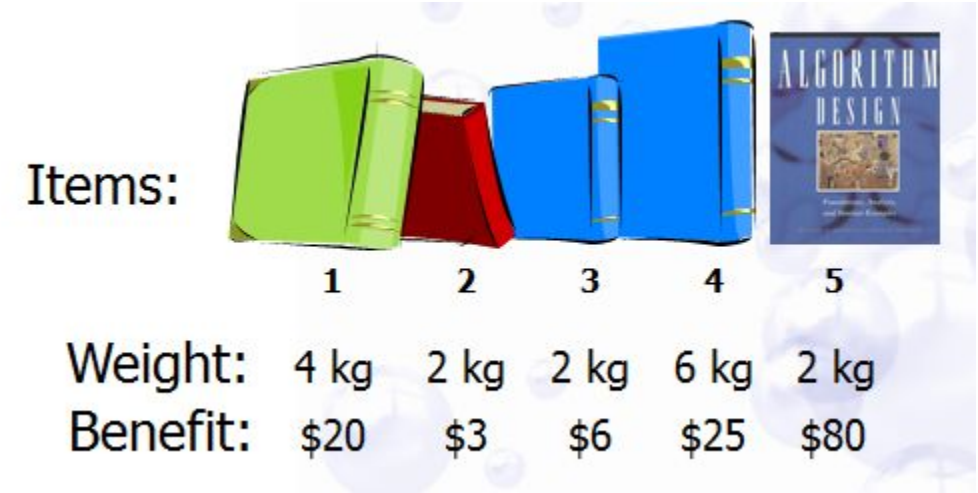
- Testing in competitive programming (Topic of group 7): use brute force in finding the correct answer for each generated test case.

- Creating unique tests for software product.

Use brute force to solve problem which have no optimal solution.

Homework

- Given: A set of 5 books, with each item i having
  - b_i - a positive benefit
  - w_i - a positive weight
- Goal: Choose books with maximum total benefit but with weight at most 8 kg (brute force only).



Reference

Brute-force search: https://en.wikipedia.org/wiki/Brute-force_search

Brute Force Attack: https://www.geeksforgeeks.org/brute-force-attack/

Brute-force: https://vnoi.info/wiki/translate/topcoder/Planning-an-Approach-to-a-Topcoder-Problem-Part-2#v%C3%A9t-c%E1%BA%A1n-brute-force


THANKS FOR
WATCHING