ls-nguyen-ngoc-hieu-1

November 3, 2023

Question 7: Given the set A= {1, 2, 3 ,...,n}, how many subsets of the following set.

Lời giải:

Mỗi tập con tương ứng với một dãy bit độ dài n, trong đó mỗi bit 0 nghĩa là không chọn phần tử tương ứng từ tập hợp A, và mỗi bit 1 nghĩa là chọn phần tử tương ứng.

Ví dụ, nếu tập hợp A có 3 phần tử (n = 3), thì có tổng cộng $2^3 = 8$ tập con khác nhau. Dãy bit có độ dài 3 là:

- 000: Không chọn bất kỳ phần tử nào.
- 001: Chọn phần tử thứ 3.
- 010: Chọn phần tử thứ 2.
- 011: Chọn phần tử thứ 2 và 3.
- 100: Chọn phần tử thứ 1.
- 101: Chọn phần tử thứ 1 và 3.
- 110: Chọn phần tử thứ 1 và 2.
- 111: Chọn tất cả 3 phần tử.

Với mỗi số n bất kỳ sẽ có 2^n các sâu nhị phân khác nhau có độ dài n $\Rightarrow$ có 2^n tập hợp con.

Question 8: In the vast expanse of the universe, deep within Commander USTH's space station, there lies a mysterious and complex structure known as the LEARNING SUPPORT doomsday device. This peculiar device occupies a significant portion of the station's interior, resulting in a rather unconventional layout for the work areas. These work areas are stacked in the form of a triangular pattern, and the diligent workers are assigned unique numerical IDs, commencing from the corner and extending outwards. This ID assignment follows a specific rule, which can be visualized in the following way:

|..

7 ..

4 8 ..

2 5 9 ..

1 3 6 10 ..

Each cell within this triangular structure can be identified by a set of coordinates (x, y) with 'x' denoting the distance from the vertical wall and 'y' representing the height from the ground. For

instance, a worker located at coordinates (1, 1) possesses the ID 1, while a bunny worker situated at (3, 2) holds the ID 9. Similarly, the bunny worker found at (2, 3) has the ID 8. This unique numbering scheme continues indefinitely, and Commander USTH has been continuously adding new workers to the station.

Your task is to create a program name Commander USTH, which, input the coordinates (x, y), will return the worker ID assigned to the bunny at that specific location.

Tóm tắt: Cho một bảng kích thước vô hạn được chia làm lưới ô vuông đơn vị. Các hàng của bảng được đánh số từ 1 từ trên xuống và các cột của bảng được đánh số từ 1 từ trái qua phải. Ô nằm trên giao điểm của hàng i, và cột j được gọi là ô (i, j). Người ta điền các số nguyên liên tiếp bắt đầu từ 1 vào bảng theo quy luật sau:

|1 3 6 10 ..

|2 5 9 ..

|4 8 ..

|7 ..

|..

Nhập vào 2 giá trị x, y hãy trả ra giá trị của ô (y, x).

Lời giải trâu bò

- Ta sử dụng vòng lặp để tìm giá trị ở ô (y, 1), sau đó tính giá trị ở ô (y, x).

Độ phức tạp O(x + y).

Python

```
[ ]: x, y = map(int, input().split())
su1 = 1

# tim gia tri o o(y, 1)
for i in range(1, y):
  su1 += i
su2 = 0
# tim gia tri o o (y, x)
for i in range(y + 1, x + y):
  su2 += i

print(su1 + su2)
```

C++

```
[ ]: #include <bits/stdc++.h>
using namespace std;
#define ll long long
signed main() {
    ll x, y;
    cin >> x >> y;
```

```
    ll su1 = 1;
    for (ll i = 1; i < y; i++)
        su1 += i;
    ll su2 = 0;
    for (ll i = y + 1; i < x + y; i++)
        su2 += i;
    cout << su1 + su2;
}
```

Lời giải cải tiến

Thay vì sử dụng vòng lặp để tính toán giá trị của ô (y, 1) và ô (y, x) ta có thể tính toán như sau.

- Kết quả vòng lặp thứ nhất (su1) bằng:

$$su1 = 1 + 1 + 2 + 3 + ... + (y-1) = 1 + \frac{y \times (y-1)}{2}$$

- Kết quả vòng lặp thứ 2 (su2) bằng:

$$su2 = y{+}1{+}y{+}2{+}....{+}...{+}(x{+}y{-}1) = y{\times}(x{-}1){+}\frac{((x-1)+1) \times (x-1)}{2} = y{\times}(x{-}1){+}\frac{(x) \times (x-1)}{2}$$

Vậy kết quả của bài toán được tính bằng công thức:

$$su1 + su2 = 1 + \frac{y \times (y-1)}{2} + y \times (x-1) + \frac{x \times (x-1)}{2}$$

Độ phức tạp tổng quát O(1)

python

```
[ ]: x, y = map(int, input().split())

print(1 + (y * (y - 1) // 2) + y * (x - 1) + x * (x - 1) // 2)
```

Cpp

```
[ ]: #include <bits/stdc++.h>
using namespace std;
#define ll long long
signed main() {
    ll x, y;
    cin >> x >> y;
    cout << 1 + (y * (y - 1) / 2) + y * (x - 1) + x * (x - 1) / 2;
}
```

Question 9: A company uses XOR checksums to create security checkpoints for checkers. You can see how the security system works in these examples:

If the third worker in line has an ID of 0 and the security checkpoint line holds three workers, the process would look like this:

0 1 2 /

3 4 / 5

6 / 7 8

The trainers XOR checksum is calculated as: $0 \hat{} 1 \hat{} 2 \hat{} 3 \hat{} 4 \hat{} 6 == 3$.

Likewise, if the first worker has an ID of 17 and the checkpoint holds four workers, the process would look like this:

17 18 19 20 /

21 22 23 / 24

25 26 / 27 28

29 / 30 31 32

This produces the checksum: $17 \hat{} 18 \hat{} 19 \hat{} 20 \hat{} 21 \hat{} 22 \hat{} 23 \hat{} 25 \hat{} 26 \hat{} 29 == 14$.

All worker IDs (including the first worker) fall between 0 and 2000000000 inclusive, and the checkpoint line will always be at least one worker long.

With this information, you can write a program that takes two values, "start" and "length," to generate the proper XOR checksum. You may need to find more information about XOR checksum if you are not already familiar with it.

Lời giải

- Ta gọi:
  - x: Biến này sẽ giảm dần và được sử dụng để theo dõi độ dài của mỗi hàng.
  - res: Biến này lưu giá trị hiện tại trong dãy.
  - lis: Là một danh sách (list) để lưu trữ tất cả các giá trị trong dãy.
- Vòng lặp bên ngoài (i) chạy từ 0 đến "length - 1." Nó sẽ theo dõi hàng hiện tại trong dãy.
- Vòng lặp bên trong (j) chạy từ 0 đến "length - 1." Nó sẽ thêm giá trị vào danh sách "lis" trong hàng hiện tại.
- ans được sử dụng để đếm các phần tử đã được thêm vào danh sách "lis" trong hàng hiện tại. Nếu ans nhỏ hơn hoặc bằng "x," nghĩa là chúng ta vẫn còn giá trị để thêm vào hàng hiện tại, và chúng ta thêm giá trị "res" vào danh sách "lis."
- Sau khi hoàn thành vòng lặp bên trong, chúng ta giảm giá trị "x" đi 1 (x -= 1) để theo dõi độ dài của hàng tiếp theo.

Cuối cùng tính kết quả xor tất cả phần tửu trong lis và tin ra kết quả.

Python

```
[ ]: start, length = map(int, input().split())

     x, res, lis = length, start, []
```

```
for i in range(length):
    ans = 0
    for j in range(length):
        ans += 1
        if(ans <= x):
            lis.append(res)
        res += 1
    x -= 1
res = lis[0]

for i in range(1, len(lis)):
    res ^= lis[i]
print(res)
```

Cpp

[ ]:
```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
signed main() {
    ll start, length;
    cin >> start >> length;

    ll x = length;
    ll res = start;
    vector<ll> lis;

    for (ll i = 0; i < length; i++) {
        ll ans = 0;
        for (ll j = 0; j < length; j++) {
            ans += 1;
            if (ans <= x) {
                lis.push_back(res);
            }
            res += 1;
        }
        x -= 1;
    }

    ll result = lis[0];
    for (ll i = 1; i < lis.size(); i++) {
        result ^= lis[i];
    }
    cout << result;
}
```

Exercise 10: Pi number can be calculated using the following formula:

$$\pi = \cfrac{4}{1+\cfrac{1}{3+\cfrac{4}{5+\cfrac{9}{7+\cfrac{16}{\cdots}}}}}$$

Write a program to calculate the value of using the above formula. (Hint: Recursion)

Input: Number of sum operations in the denominator

Output: Estimated number (rounded to 5 digits after the decimal point)

Example:

If the input is 2, then the output is:

$$\pi = \cfrac{4}{1+\cfrac{1}{3+4}}$$

Lời giải

Để bài toán trở lên đơn giản ta sẽ tính kết quả từ dưới lên trên bằng cách sử dụng vòng lặp.

- Với mỗi gạch ngang ở phép chia ta sẽ coi là một hàng

Vậy hàng cuối sẽ được tính bằng công thức: (n * 2) - 1 + x.

sau khi biết được các chỉ số của hàng cuối ta cứ tính lên trên dần và cuối cùng chỉ cần lấy 4 cho giá trị vừa tính được sẽ in ra được kết quả.

Độ phức tạp tổng quát: O(n). Với đpt này ta hoàn toàn có thể ăn được điểm super bunus point.

Code python

[ ]:
```python
n = int(input())
x, k = 1, 3
for i in range(n - 1):
    x = x + k;
    k += 2
k -= 2
kq = x + k
for i in range(n - 1):
    kq = (k - 2) + (x - k) / kq;
    k, x = k - 2, x - k
kq = 4 / kq
print('{0:.{1}f}'.format(kq, 5))
```

Code cpp

[ ]:
```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
signed  main() {
    ll n; cin >> n;
    ll x = 1, k = 3;
```

```
    for (int i = 0; i < n - 1; i++) {
        x = x + k;
        k += 2;
    }
    k -= 2;
    double kq = x + (k * 1.0);
    for (int i = 0; i < n - 1; i++) {
        kq = (k - 2) + ((x - k) / (kq * 1.0));
        x -= k;
        k -= 2;
    }
    cout << fixed << setprecision(5) << 4 / kq;
}
```