1. Técnicas de programación

Angel Napa

March 31, 2023

Resumen

Introducción

Language Features

Iteratively complete search

Recursive complete search

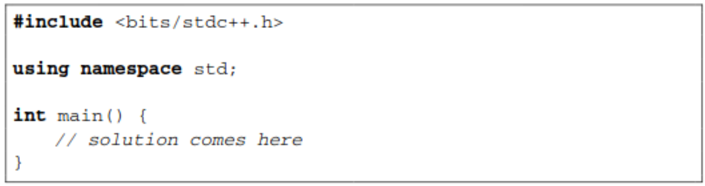
Introducción

Introduccion

- We will review some language Features in C++
- Introduction to basic search algorithms, in particular two of usual types of complete search: iterative and recursive complete search

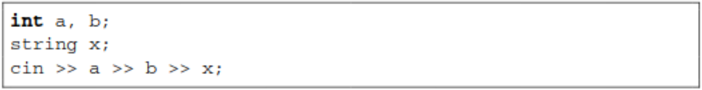
Language Features

Template

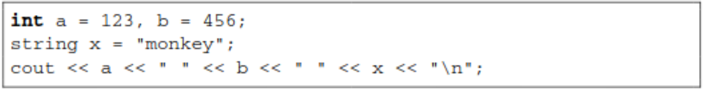


```
#include <bits/stdc++.h>

using namespace std;

int main() {
    // solution comes here
}
```

Input & Output

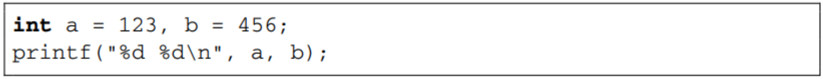


```
int a, b;
string x;
cin >> a >> b >> x;
```

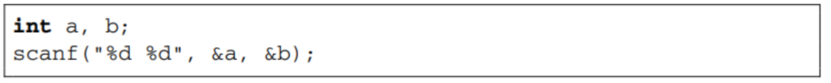


```
int a = 123, b = 456;
string x = "monkey";
cout << a << " " << b << " " << x << "\n";
```

Input & Output



```
int a = 123, b = 456;
printf("%d %d\n", a, b);
```



```
int a, b;
scanf("%d %d", &a, &b);
```

Data Type

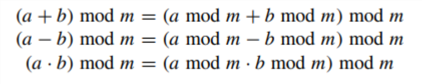
- These are the basic data types:
  - **bool**: a boolean (`true`/`false`)
  - **char**: an 8-bit signed integer (ASCII)
  - **short**: a 16-bit signed integer
  - **int**: a 32-bit signed integer
  - **long long**: a 64-bit signed integer
  - **float**: a 32-bit floating-point number
  - **double**: a 64-bit floating-point number
  - **long double**: a 128-bit floating-point number
  - **string**: a string of characters

Data type

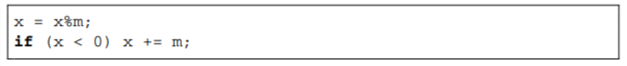
- $10^3 \approx 2^{10}$
- $int \in [-2^{31}, 2^{31}[\approx [-2 \cdot 10^9, 2 \cdot 10^9[$
- long long $\in [-2^{63}, 2^{63}[\approx [-9.2 \cdot 10^{18}, 2 \cdot 10^{18}[$
- float $\in I \approx [-3.4 \cdot 10^{38}, 3.4 \cdot 10^{38}]$
- precision of float ≈ 7 decimal digits
- double $\in I \approx [-1.7 \cdot 10^{308}, 1.7 \cdot 10^{308}]$
- precision of double ≈ 14 decimal digits
- long double $\in I \approx [-1.1 \cdot 10^{4932}, 1.1 \cdot 10^{4932}]$
- precision of double ≈ 18 decimal digits

Modular Arithmetic

- Remember operator %
- Some problems only ask for the remainder of certain operation given some module
- Be careful if the number is negative, since its module will be also negative



$$(a + b) \bmod m = (a \bmod m + b \bmod m) \bmod m$$
$$(a - b) \bmod m = (a \bmod m - b \bmod m) \bmod m$$
$$(a \cdot b) \bmod m = (a \bmod m \cdot b \bmod m) \bmod m$$

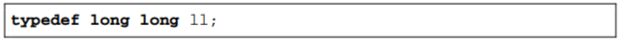


```
x = x%m;
if (x < 0) x += m;
```

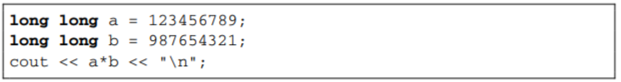
Shortening Code

- sometimes it is useful to simplify standard expressions
- We can do this thanks to the Type names and Macros
- Typenames rename data types, while macros rename certain strings in the code

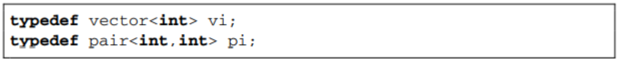
Shortening Code



```
typedef long long ll;
```

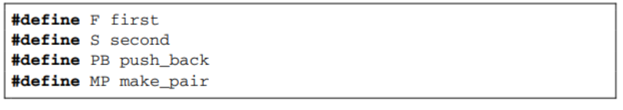


```
long long a = 123456789;
long long b = 987654321;
cout << a*b << "\n";
```

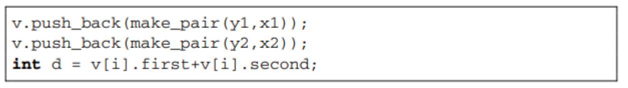


```
typedef vector<int> vi;
typedef pair<int,int> pi;
```

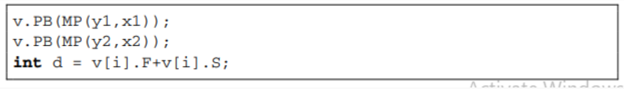
Shortening Code



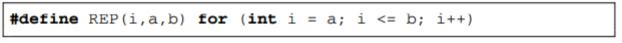
```
#define F first
#define S second
#define PB push_back
#define MP make_pair
```



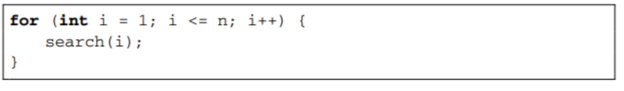
```
v.push_back(make_pair(y1,x1));
v.push_back(make_pair(y2,x2));
int d = v[i].first+v[i].second;
```



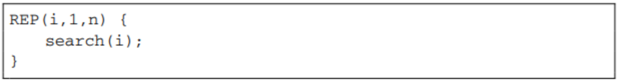
```
v.PB(MP(y1,x1));
v.PB(MP(y2,x2));
int d = v[i].F+v[i].S;
```



```
#define REP(i,a,b) for (int i = a; i <= b; i++)
```



```
for (int i = 1; i <= n; i++) {
    search(i);
}
```



```
REP(i,1,n) {
    search(i);
}
```

Complete search

- Develop a Complete Search solution when there is clearly no other algorithm available
- A Complete Search solution may receive a Time Limit Exceeded (TLE) verdict
- We should do a proper analysis before attempting to code
- Iteratively vs recursively (backtracking)

Iteratively complete search

Iteratively Complete Search

Two Nested Loop

- UVa 725 - Division
- Find and display all pairs of 5-digit numbers that collectively use the digits 0 through 9 once each
- abcde / fghij = N, where each letter represents a different digit
- fghij can only range from 01234 to 98765 which is at most 100K

Two Nested Loop

- UVa 725 - Division
- Find and display all pairs of 5-digit numbers that collectively use the digits 0 through 9 once each
- abcde / fghij = N, where each letter represents a different digit

Two Nested Loop

- fghij can only range from 01234 to 98765 which is at most 100K
- Better bound for fghij is the range 01234 to 98765 / N
- For each attempted fghij, we can get abcde from fghij * N and then check if all 10 digits are different
- Doubly-nested loop with a time complexity of at most 50K $\times 10 = 500$K

Many Nested Loops

- UVa 441 - Lotto
- Given $6 < k < 13$ integers, enumerate all possible subsets of size 6 of these integers in sorted order.
- Required subset is always 6
- Output has to be sorted lexico-graphically
- Input is already sorted

Many Nested Loops

- Easiest solution is to use six nested loops
- Even in the largest test case when k = 12, these six nested loops will only produce $\binom{12}{6} = 924$

Loops + Pruning

- UVa 11565 - Simple Equations
- Given three integers A, B, and C ($1 \leq A, B, C \leq 10000$), find three other distinct integers x, y, and z such that $x + y + z = A, xyz = B$, and $x^2 + y^2 + z^2 = C$

Loops + Pruning

- Observe equation $x^2 + y^2 + z^2 = C$
- As $C \leq 10000$, we must have $-100 \leq x \leq 100$.
- Analogously, we must have $-100 \leq y, z \leq 100$.
- We can then write the a triply-nested iterative solution below that requires $201 \times 201 \times 201 \approx 8M$

Bitmasking

- Uva 12455 - Bars
- Given a list l containing $1 \leq n \leq 20$ integers, is there a subset of list l that sums to another given integer X
- We can try all 2^n possible subsets of integers, sum the selected integers for each subset in $O(n)$, and see if the sum of the selected integers equals to X
- Overall time complexity is thus $O(n \cdot 2^n)$
- When $n = 20$, this is just $20 \times 2^{20} \sim 21M$

Recursive complete search

Simple Backtracking

- UVa 750 - 8 Queens Chess Problem
- In chess (with an 8 × 8 board), it is possible to place eight queens on the board such that no two queens attack each other.
- Determine all such possible arrangements given the position of one of the queen

Gracias