TWGSS OI Team

2022 Mini Contest II

Editorial

Word Connect

TC222I

Diskalmer

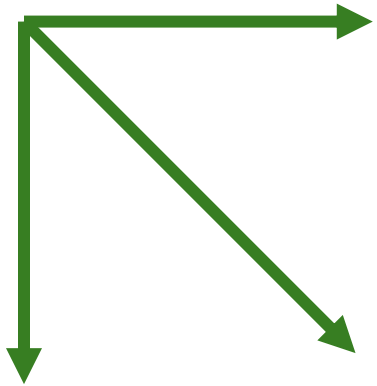
word CONNECT

This problem have nothing to do with “connect”, more like disconnect lol.

Problem Statement

Given a grid and a string, calculate the occurrence of the string in the grid in 3 directions:

1. Left to right
2. Top to bottom
3. Top left to bottom right



Input Format

Input is a $N \times M$ grid and a string B.

2 approach to store the grid:

1. Use 1D array of `string`
2. Use 2D array of `char`

(Some participants used a string to store the whole array, which may cause error later on.)

Use `string` to store string B.

Subtasks

1. Only consider one line
2. Only consider horizontal
3. Only consider horizontal and vertical
4. Full solution

Approach for Subtask 1

$N = 1$

Anyone have idea?

For each character in the grid, we let it be the "starting char". Compare the following characters and check if the characters are the same with the target string.

Subtask 1 Solution

```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){
  int n,m,sh,sv,sd,k;
  cin >> n >> m >> sh >> sv >> sd;
  string S,B;
  cin >> S;
  cin >> k;
  cin >> B;
  int ans=0;
  bool same;
  for (int i=0;i<m;i++){//let each char be "head char"
    same = 1;
    if (i+k>m){//char after this will exceed the boundary
      break;
    }
    for (int j=0;j<k;j++){//check if the substring is identical
      if (B[j] != S[i+j]) {
        //if this is true means it is not identical
        same = 0;
        break;
      }
    }
    if (same){
      ans+=sh;
    }
  }
  cout << ans << endl;
  return 0;
}
```

Time complexity: O(MK)

Score: 10

You could use `substr()` function too.

Subtask 1 Solution

I have the same idea, why do I get Wrong Answer / Runtime Error?

So sad. Maybe you access the grid out-of-bounds.

Huh? How can I fix it?

You have to check for the boundary.

If `i+K>M` → `break` it!

Or you can just do restrictions in the condition of the `for` loop.

Approach for Subtask 2

$s_h = s_d = 0$

i.e. Only need to consider horizontal matches

VERY similar to subtask 1.

For each line, you do the same thing as subtask 1.

Subtask 2 Solution

```cpp
#include<bits/stdc++.h>
using namespace std;

string S[501];
int main(){
  int n,m,sh,sv,sd,k;
  cin >> n >> m >> sh >> sv >> sd;
  string B;
  for (int i=0;i<n;i++){
    cin >> S[i];
  }
  cin >> k;
  cin >> B;
  int ans=0;
  bool same;
  for (int h=0;h<n;h++){
    for (int i=0;i<m;i++){
      same = 1;
      if (i+k>m){
        break;
      }
      for (int j=0;j<k;j++){
        if (B[j] != S[h][i+j]) {
          same = 0;
          break;
        }
      }
      if (same) ans+=sh;
  }
  cout << ans << endl;
  return 0;
}
```

Time complexity: O(NMK)

Score: 5 (Cumulative: 15)

Approach for Subtask 3

$s_d = 0$

i.e. Only need to consider horizontal and vertical matches

You need to check for vertical too now

If you can do subtask 2, you just need to make some minor changes

Swap the dimensions of the `for` loop

→ for each column, check if the strings is the same from top to bottom

Subtask 3 solution

```cpp
#include<bits/stdc++.h>
using namespace std;
#define int long long
char grid[5001][5001];
int32_t main(){
  int n,m,sh,sv,sd;
  cin >> n >> m >> sh >> sv >> sd;
  for (int i=0;i<n;i++){
    for (int j=0;j<m;j++){
      cin >> grid[i][j];
    }
  }
  int K;
  string S;
  cin >> K >> S;
  int ans = 0;
  bool same;
  for (int h=0;h<n;h++){
    for (int i=0;i<=m;i++){
      same = 1;
      if (i+K>m){
        break;
      }
      for (int j=0;j<K;j++){
        if (grid[h][i+j] != S[j]){
          same = 0;
          break;
        }
      }
      if (same) ans+=sh;
    }
  }
  for (int h=0;h<m;h++){//for each column
    for (int j=0;j<=n;j++){//for each column in column
      same = 1;
      if (j+K>n){//exceed boundary
        break;
      }
      for (int i=0;i<K;i++){
        if (grid[j+i][h] != S[i]){//be aware i+j and h is swapped
          same = 0;
          break;
        }
      }
      if (same) ans+=sv;
    }
  }
  cout << ans << endl;
  return 0;
}
```

Time complexity: O(NMK)

Score: 5 (Cumulative: 20)

Approach for Full Solution

Similar concept but a bit harder.

Both row and column need to be changed when checking.

Be careful that there are one more perimeter to check exceed boundary or not.

Full Solution

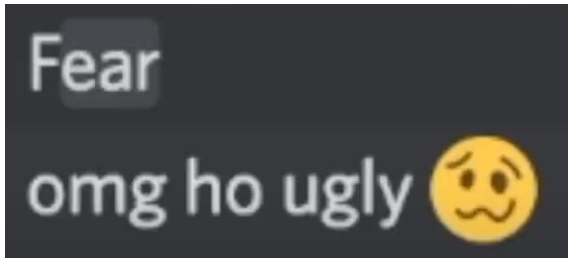
```cpp
#include<bits/stdc++.h>
using namespace std;

int main(){
  cout << "Hello World!" << endl;
  return 0;
}
```

The full solution is left as an exercise for the reader :)

Time complexity: O(NMK)

Score: 25



Starviet Union

TC2222

Basic Problem Setting

Given N numbers, find a longest sequence within these numbers which is strictly increasing.

(= if the brightness of the stars are followed by an ascending order, Tom will be happy!)

FYI: (Ascending Order)

1, 3, 5, 7, 9 (1 < 3 < 5 < 7 < 9)

3, 6, 8, 19, 40 (3 < 6 < 8 < 19 < 40)

Basic Problem Setting

However, we cannot change the order of numbers to fulfil the sequence.

What we could do is only counting.

Examples:

5
8 9 3 6 4
Longest Increasing Subsequence in these numbers: 2
Possible Cases: 8, 9 / 3, 6 / 3, 4

6
1 5 5 8 5 9
Longest Increasing Subsequence in these numbers: 4
Possible Cases: 1, 5, 8, 9

Subtask 1

How can we do that?

For subtask 1 ($1 \leq N \leq 5000$),

We could use a naïve and "not so big brain" method:

Try all the numbers!

Subtask 1

```
for (int i = 1; i <= N; i++){
  DP[i] = 1;
  for (int j = 1; j < i; j++){
    if(a[i] > a[j]){
      DP[i] = max(DP[i], DP[j] + 1);
    }
  }
}
```

Use a double for-loop to try out all numbers

We construct another array (DP[ ]) to store the longest increasing subsequence in each index.

Subtask 1: Explanation

```
for (int i = 1; i <= N; i++){
  DP[i] = 1;
  for (int j = 1; j < i; j++){
    if(a[i] > a[j]){
      DP[i] = max(DP[i], DP[j] + 1);
    }
  }
}
```

For DP (dynamic programming):

"We get the final answer by tracing back to the confirmed answer before."

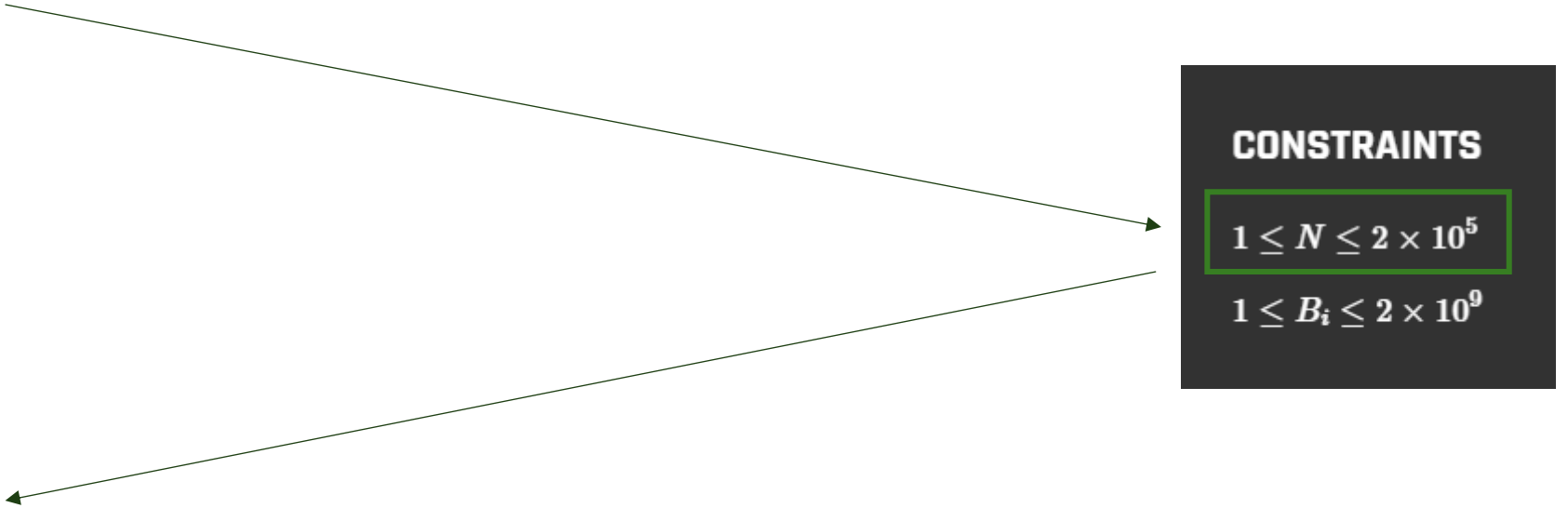
…(a lot of numbers)……A……(a lot of numbers)……B……

What can we prove in this situation?
If A < B and numbers between A and B are all > B,

Then the longest increasing subsequence by B
=
The longest increasing subsequence by A + 1

Full Solution

Time complexity of $O(N^2)$ is definitely not fast enough to fully complete this problem.



If the subtask 1 solution is used,

Operations: $(2 \times 10^5)^2 = 40000000000$!!!

Full Solution

Can we shorten the time taken for this question?

Recall back to Subtask 1, we used double for-loops to search for all possible combinations.

How can we improve that?

Full Solution

Probably we should use a vector as we could gradually construct a sequence which is followed by ascending order (using push_back).

Examples:

5
1 4 5 3 6

1st: 1 4 5 3 6
Vector: 1

2nd: 1 4 5 3 6
Vector: 1, 4

3rd: 1 4 5 3 6
Vector: 1, 4, 5

In these few steps, the vector can be directly constructed as the numbers are still increasing.
(1 < 4 < 5)

But how about the next number?
(3 < 5…)

Full Solution

We use searching and replace the old numbers for numbers smaller than the last number of the subsequence.

Examples:

5
1 4 5 3 6

4th: 1 4 5 3 6

Vector:
(1, 4, 5) → (1, 3, 5)

Purpose for this action:

We "passed through" this number, but it doesn't count any +1 for the final answer.

Full Solution

We find the smallest element which is larger than your next number inserted into the vector.

Examples(2):

5
1 6 4 5 7

1st: 1 6 4 5 7
Vector: 1

2nd: 1 6 4 5 7
Vector: 1 6

3rd : 1 6 4 5 7
Vector: (1 6) → (1 4)

4th : 1 6 4 5 7
Vector: 1 4 5

5th : 1 6 4 5 7
Vector: 1 4 5 7

Full Solution

However, if we use regular searching method (linear searching), the time complexity is still $O(N^2)$ ☹

How can we speed up?

Binary Search

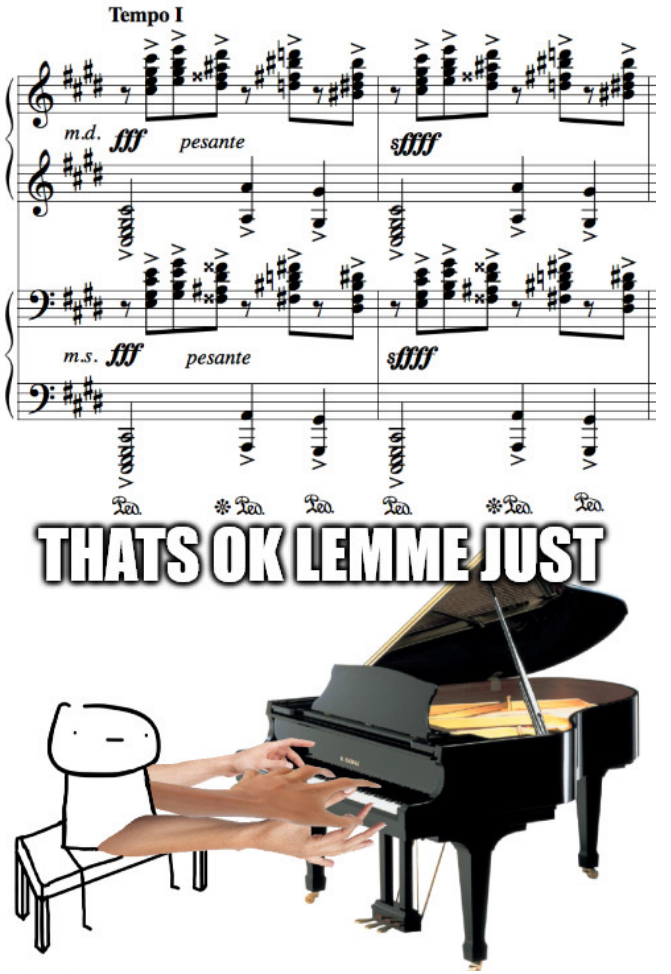
Piano Keys

TC2223

Problem Idea

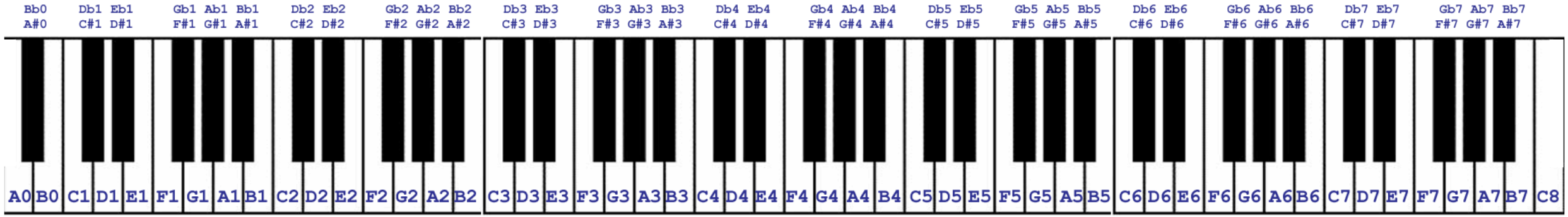
Do you know how to play piano?

It is sad if you don't know, and so do I though.



The Piano

Here's a standard 88-keys piano:



Wanna date with A0?
No, please.

Octave 0 * $A_0, A\sharp_0/B\flat_0, B_0,$

Octave 1 $C_1, C\sharp_1/D\flat_1, D_1, D\sharp_1/E\flat_1, E_1, F_1, F\sharp_1/G\flat_1, G_1, G\sharp_1/A\flat_1, A_1, A\sharp_1/B\flat_1,$

Octave 2-7 $C_2, C\sharp_2/D\flat_2, \ldots, A_7, A\sharp_7/B\flat_7,$

Octave 8 * C_8

* not a full octave

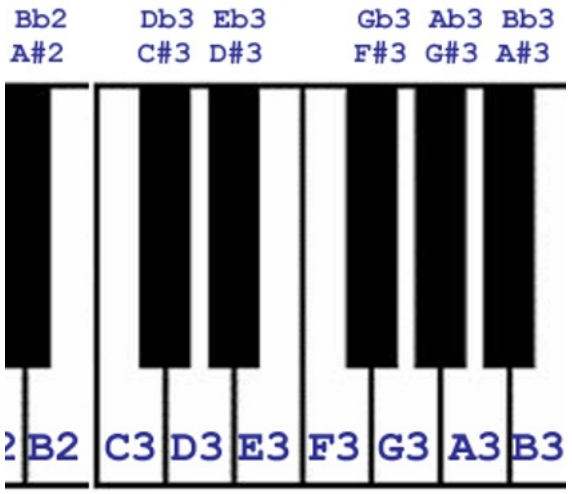
Problem Statement

You have 2 hands, having 5 fingers each.

Each hand can reach a span of D keys.

(black key also count as a key, *black key matters*!)

Example:



C_3 to G_3 → 8 keys span

C_3 to $D\sharp_3$ / $E\flat_3$ → 4 keys span

$D\sharp_3$ / $E\flat_3$ to $F\sharp_3$ / $G\flat_3$ → 4 keys span

CORONA
DisConnecting People

This is just a random meme showing that you have 2 hands with 5 fingers each

Problem Statement

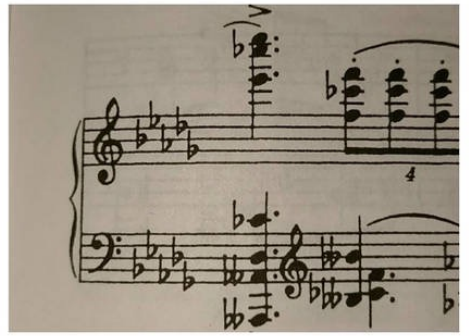
Each hand can reach a span of D keys.

There are N keys you have to press at the same time.

Target:

Find at most how many keys out of the N keys we can press using our only two hands with keys span D each.

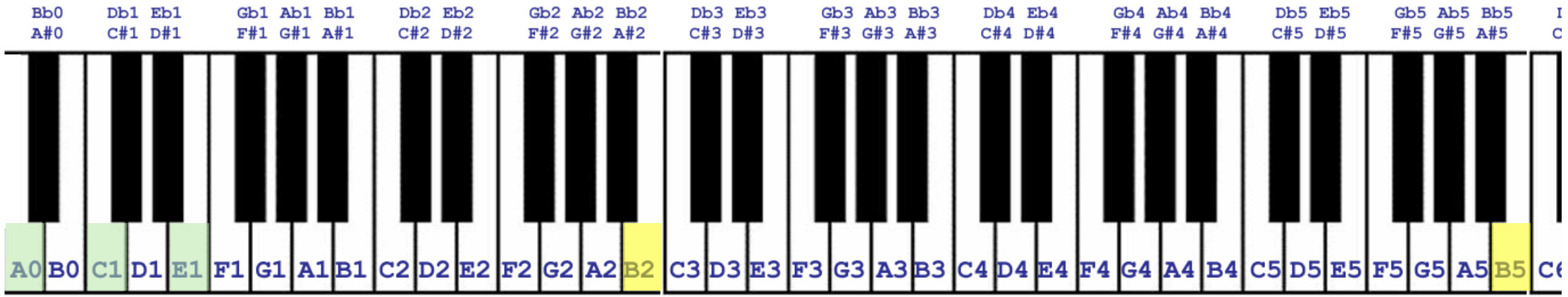
THIS CHORD:



NO PROBLEM:

Sample Tests

Input	5 8 A0 C1 E1 B2 B5
Output	4



A_0 C_1 E_1

Use 3 fingers from left hand

∵ Keys span = 8 ≤ $D(8)$

∴ **Can** press at the same time → ans +3

B_2 B_5

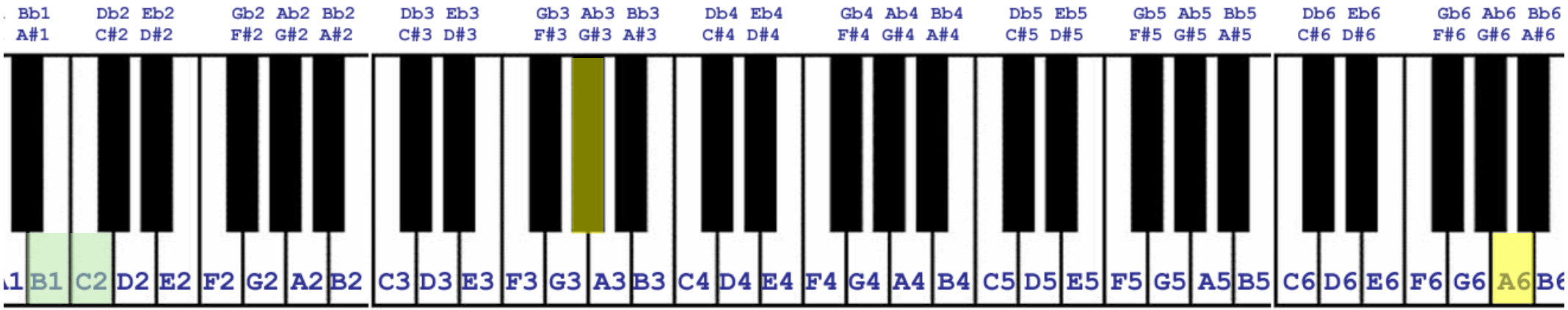
∵ Keys span = 37 > $D(8)$

∴ **Cannot** press at the same time

∴ Choose either one to press using right hand → ans +1

Sample Tests

Input	4 10 B1 C2 G#3 A6
Output	3



B_1 C_2

Use 2 fingers from left hand

∵ Keys span = 2 ≤ $D(10)$

∴ **Can** press at the same time → ans +2

(B_1 to $G\sharp_3$ → keys span: 22 > $D(10)$ → No)

$G\sharp_3$ A_6

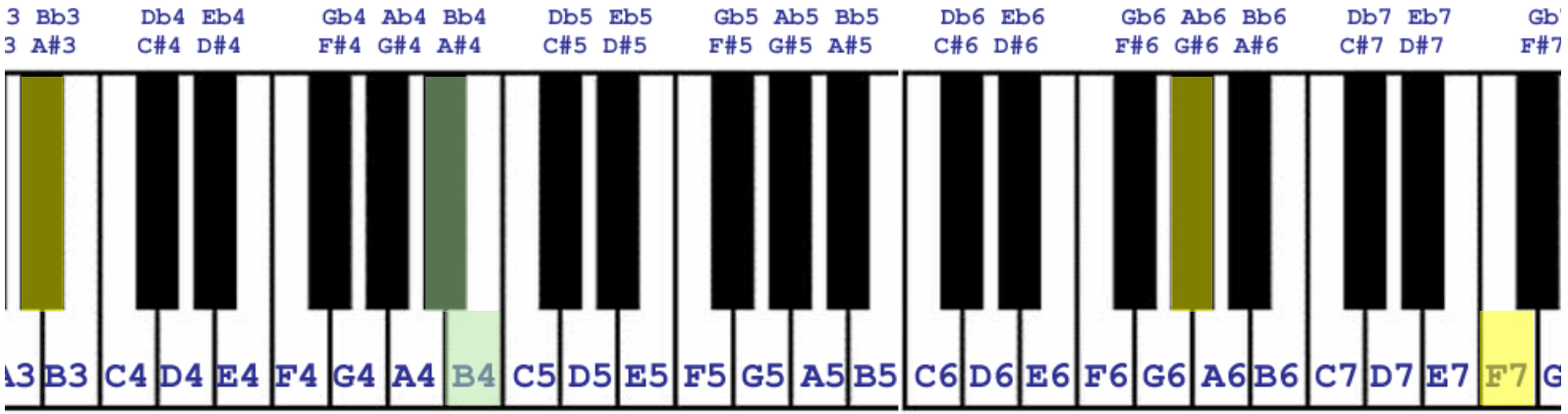
∵ Keys span = 38 > $D(10)$

∴ **Cannot** press at the same time

∴ Choose either one to press using right hand → ans +1

Sample Tests

Input	5 3 A#3 A#4 B4 Ab6 F7
Output	3



$A\sharp_4$ B_4

Use 2 fingers from one hand

∵ Keys span = 2 ≤ $D(3)$

∴ **Can** press at the same time → ans +2

$A\sharp_3$ $A\flat_6$ F_7

Cannot press at the same time

Choose either one to press using another hand → ans +1

Subtask 1…?

$1 \leq N \leq 7$

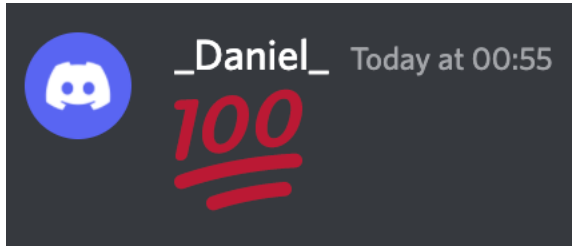
All the notes are in octave 1

There is no # or b signs in the notes

EASY! Just hard code!

There are just $2^7 \times 88 = 11264$ cases!

Just use 11264 `if` statements, and you will do the trick!



No please.

Subtask 1…?

We can reduce the number of cases by knowing the answer is N when $D \geq 6$.

Why? There are total of 12 keys in octave 1, when $D \geq 6$, all keys are covered.

New no. of cases is $2^7 \times 5 + 1 = 641$.

Wow! Much easier!

Or is it worth it spending that much time writing hundreds of `if` statements?

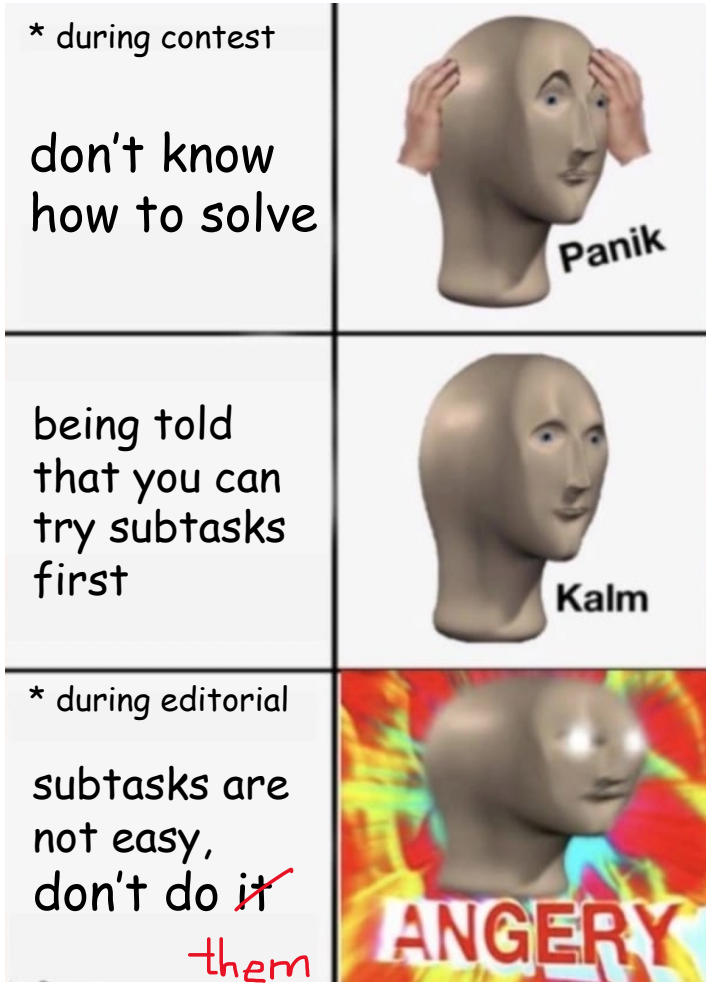
Absolutely not!

Therefore, we'll skip to…

Full Solution

What?!?! Skip straight to the full solution???

Actually the subtasks are not easy, they are just for saving you from not getting a 0 mark when you full solution have some implementation fault or don't know how to implement part of the solution.



The Idea

What can the time complexity be?

Can it be $O(N^2)$?

Of course!

$N^2 = 88^2 = 7744$

Such a small number!

But… how to achieve it?

The Idea

The **outer** loop (left hand): scanning the required keys from left to right

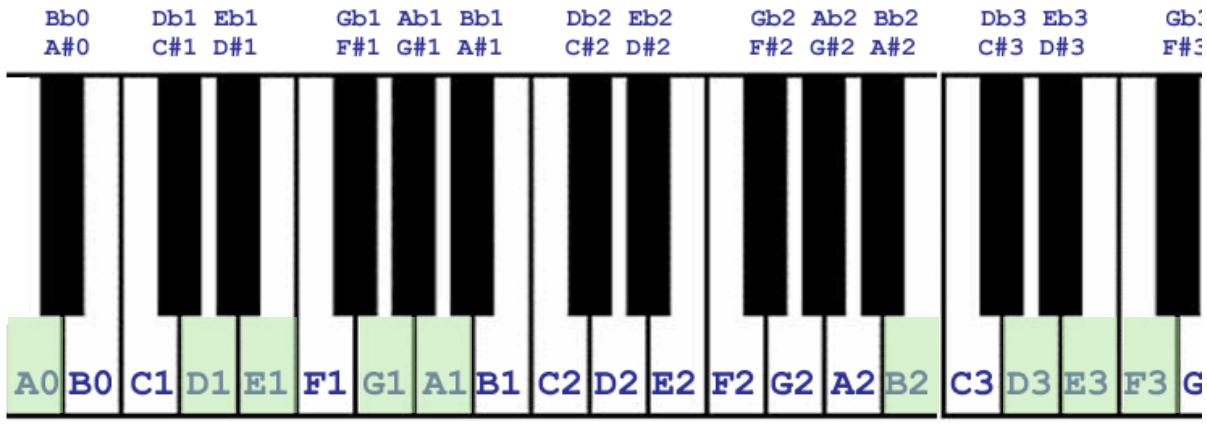
For each key scanned, check the furthest (rightmost) key it can reach with one hand starting from it.

The **inner** loop (right hand): scanning the required keys starting form the key after the furthest key we get from the outer loop.

For each key scanned, check the furthest (rightmost) key it can reach with one hand starting from it.

Visualisation

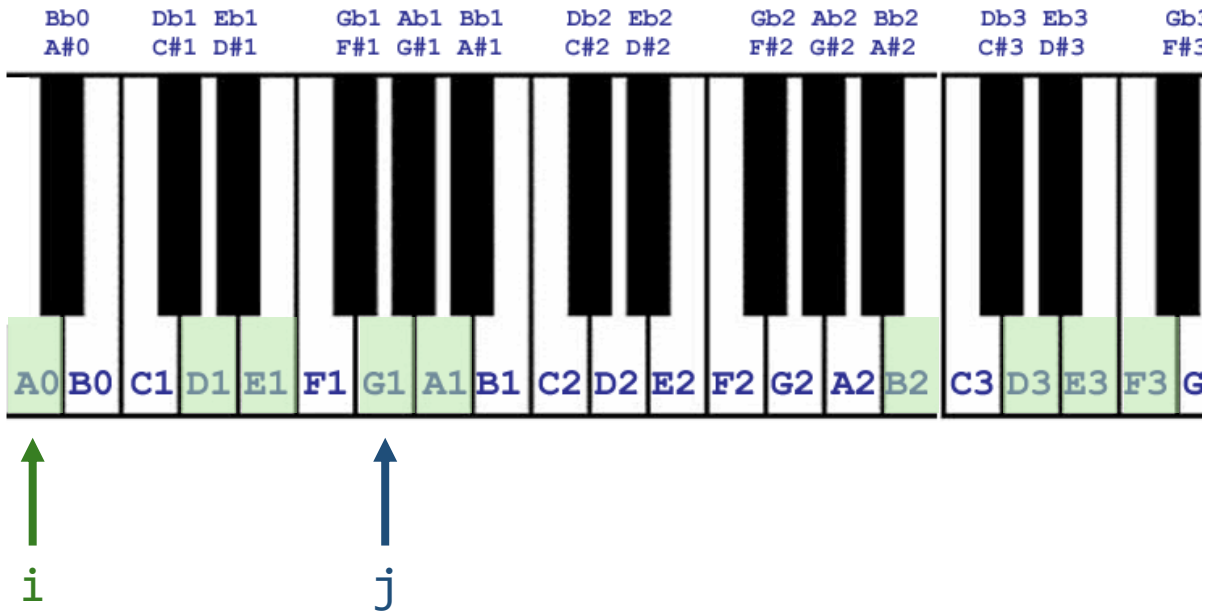
Input	9 8 A0 D1 E1 G1 A1 B2 D3 E3 F3



Outer loop	Position (i)	
	Furthest	
	Key pressed	
Inner loop	Position (j)	
	Furthest	
	Key pressed	
Sum of this combination		
Current answer		0

Visualisation

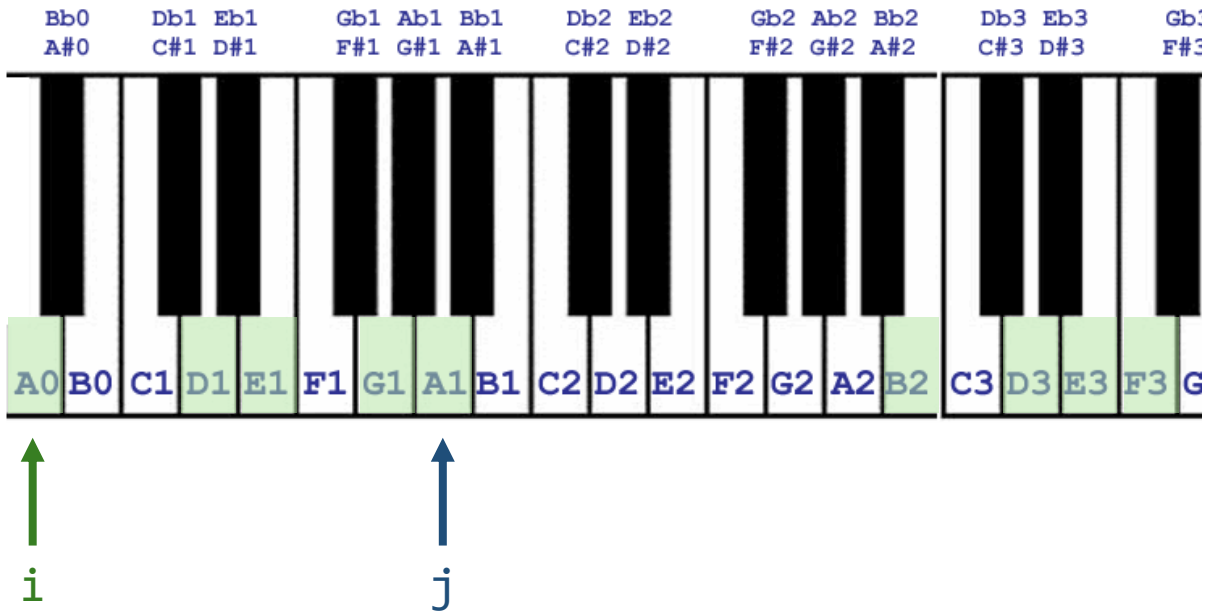
Input	9 8 A0 D1 E1 G1 A1 B2 D3 E3 F3



Outer loop	Position (i)	A0
	Furthest	E1
	Key pressed	3
Inner loop	Position (j)	G1
	Furthest	A1
	Key pressed	2
Sum of this combination		5
Current answer		0 → 5

Visualisation

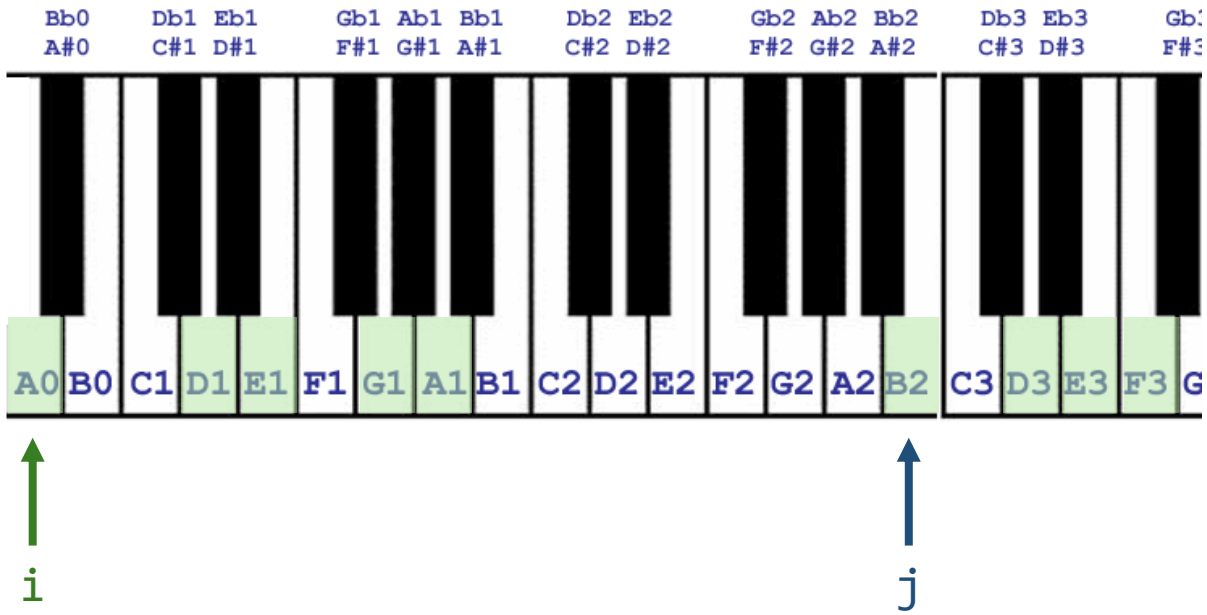
Input	9 8 A0 D1 E1 G1 A1 B2 D3 E3 F3



Outer loop	Position (i)	A0
	Furthest	E1
	Key pressed	3
Inner loop	Position (j)	A1
	Furthest	A1
	Key pressed	1
Sum of this combination		4
Current answer		5

Visualisation

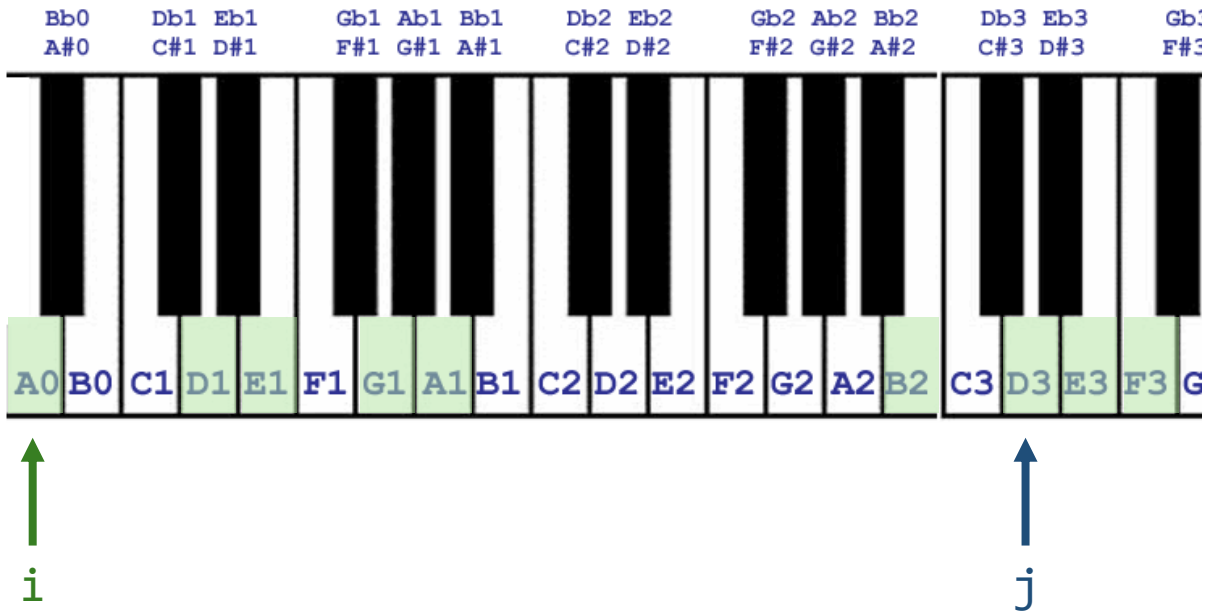
Input	9 8 A0 D1 E1 G1 A1 B2 D3 E3 F3



Outer loop	Position (i)	A0
	Furthest	E1
	Key pressed	3
Inner loop	Position (j)	B2
	Furthest	F3
	Key pressed	4
Sum of this combination		7
Current answer		5 → 7

Visualisation

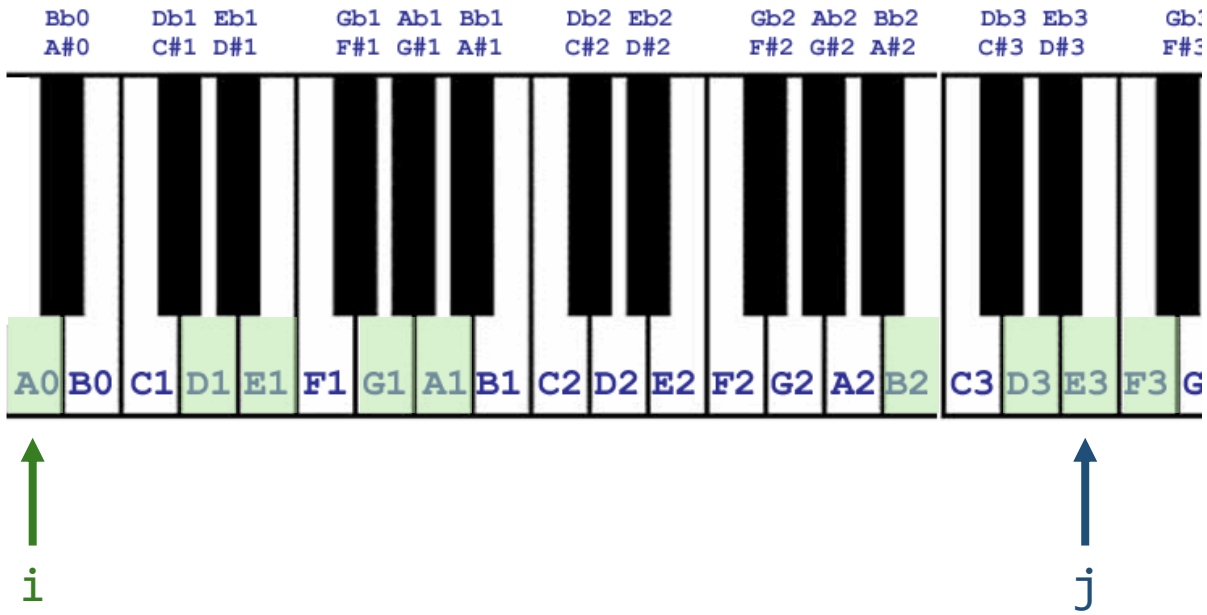
Input	9 8 A0 D1 E1 G1 A1 B2 D3 E3 F3



Outer loop	Position (i)	A0
	Furthest	E1
	Key pressed	3
Inner loop	Position (j)	D3
	Furthest	F3
	Key pressed	3
Sum of this combination		6
Current answer		7

Visualisation

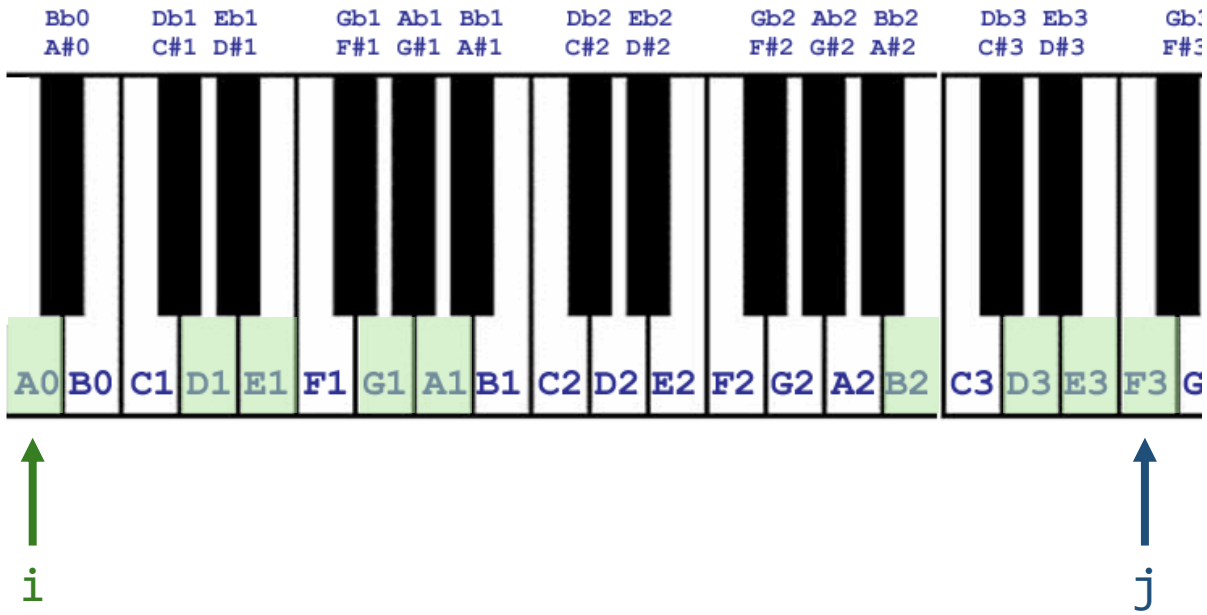
Input	9 8 A0 D1 E1 G1 A1 B2 D3 E3 F3



Outer loop	Position (i)	A0
	Furthest	E1
	Key pressed	3
Inner loop	Position (j)	E3
	Furthest	F3
	Key pressed	2
Sum of this combination		5
Current answer		7

Visualisation

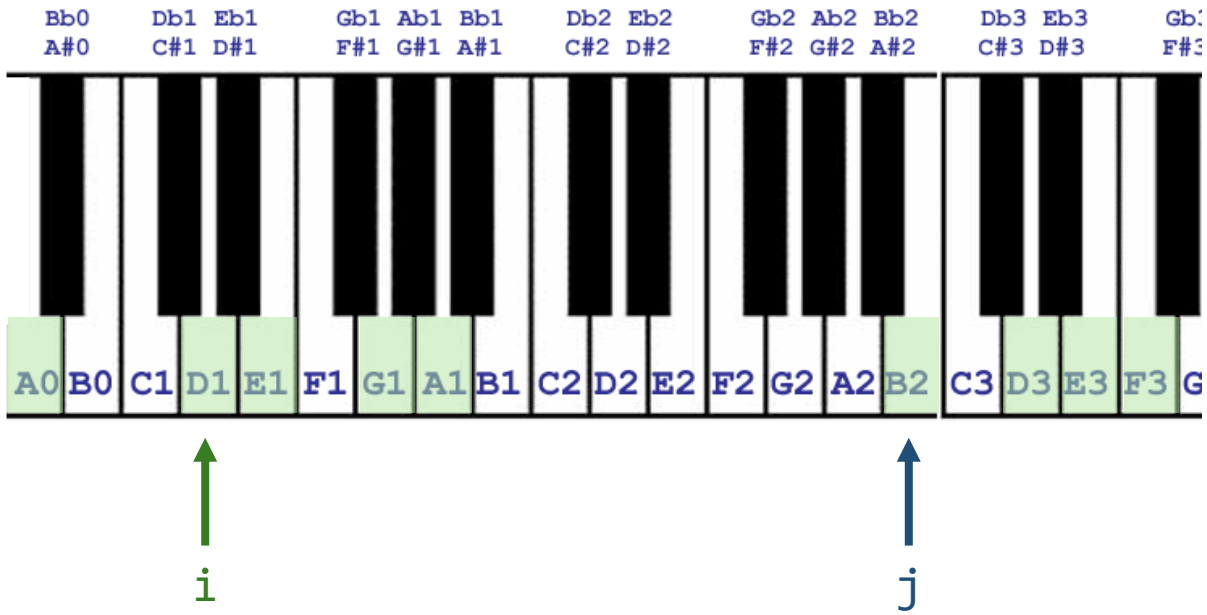
Input	9 8 A0 D1 E1 G1 A1 B2 D3 E3 F3



Outer loop	Position (i)	A0
	Furthest	E1
	Key pressed	3
Inner loop	Position (j)	F3
	Furthest	F3
	Key pressed	1
Sum of this combination		4
Current answer		7

Visualisation

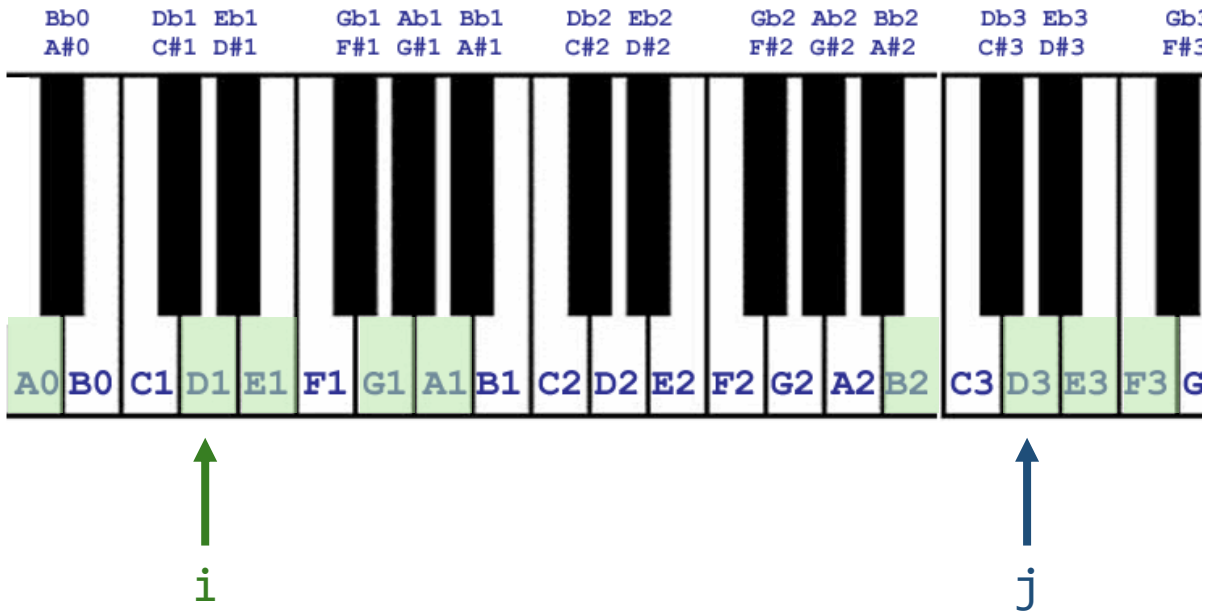
Input	9 8 A0 D1 E1 G1 A1 B2 D3 E3 F3



Outer loop	Position (i)	D1
	Furthest	A1
	Key pressed	4
Inner loop	Position (j)	B2
	Furthest	F3
	Key pressed	4
Sum of this combination		8
Current answer		7 → 8

Visualisation

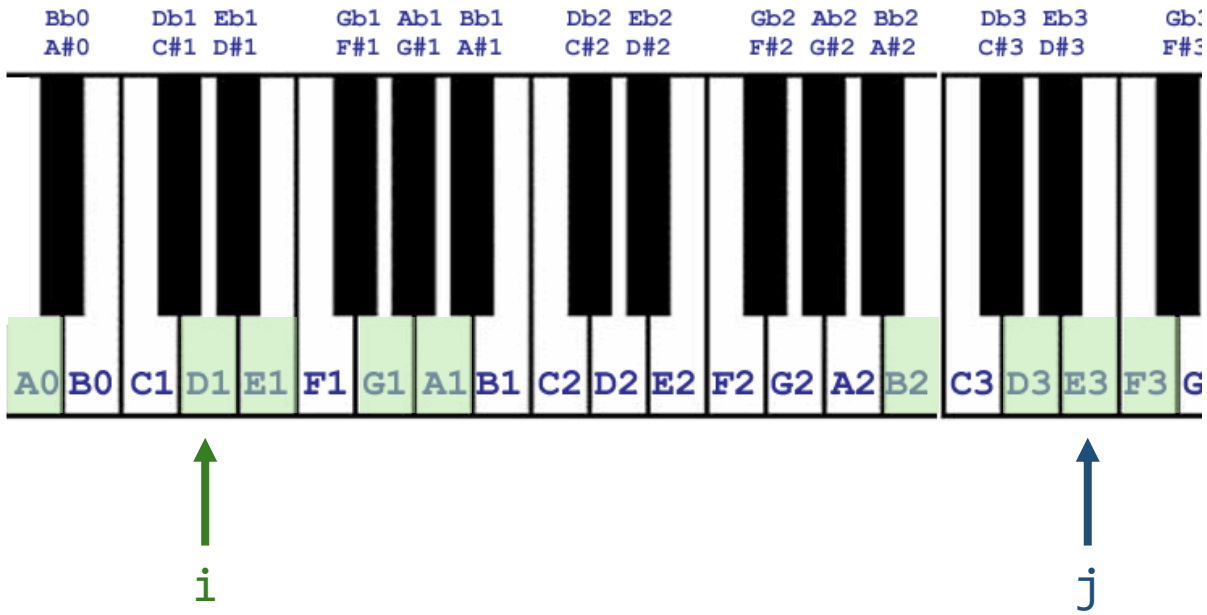
Input	9 8 A0 D1 E1 G1 A1 B2 D3 E3 F3



Outer loop	Position (i)	D1
	Furthest	A1
	Key pressed	4
Inner loop	Position (j)	D3
	Furthest	F3
	Key pressed	3
Sum of this combination		7
Current answer		8

Visualisation

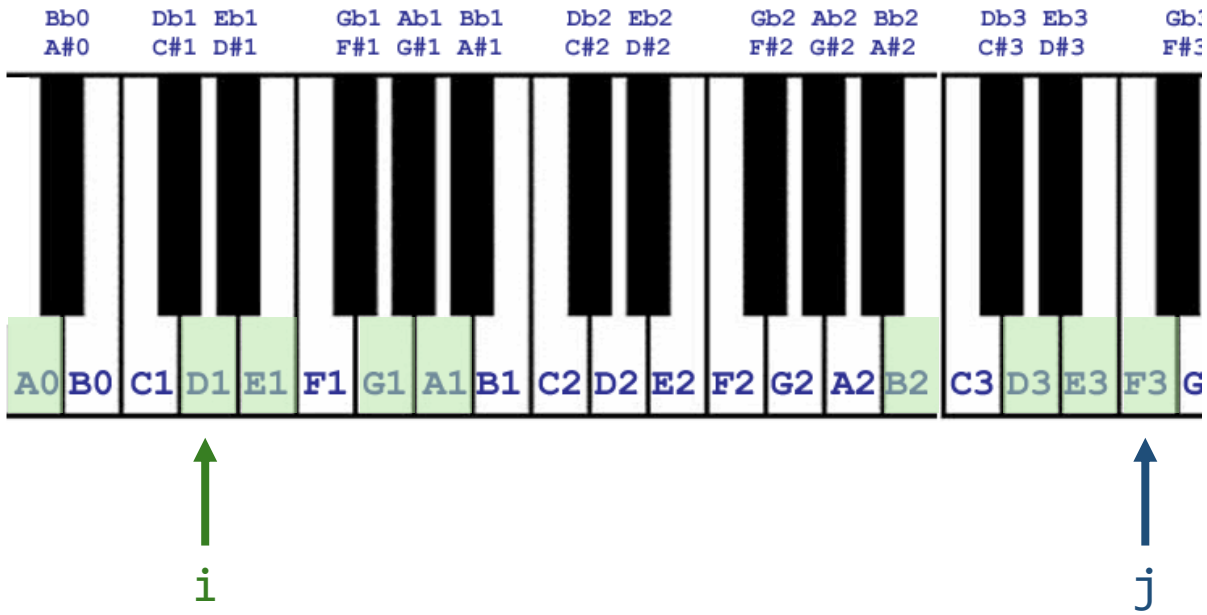
Input	9 8 A0 D1 E1 G1 A1 B2 D3 E3 F3



Outer loop	Position (i)	D1
	Furthest	A1
	Key pressed	4
Inner loop	Position (j)	E3
	Furthest	F3
	Key pressed	2
Sum of this combination		6
Current answer		8

Visualisation

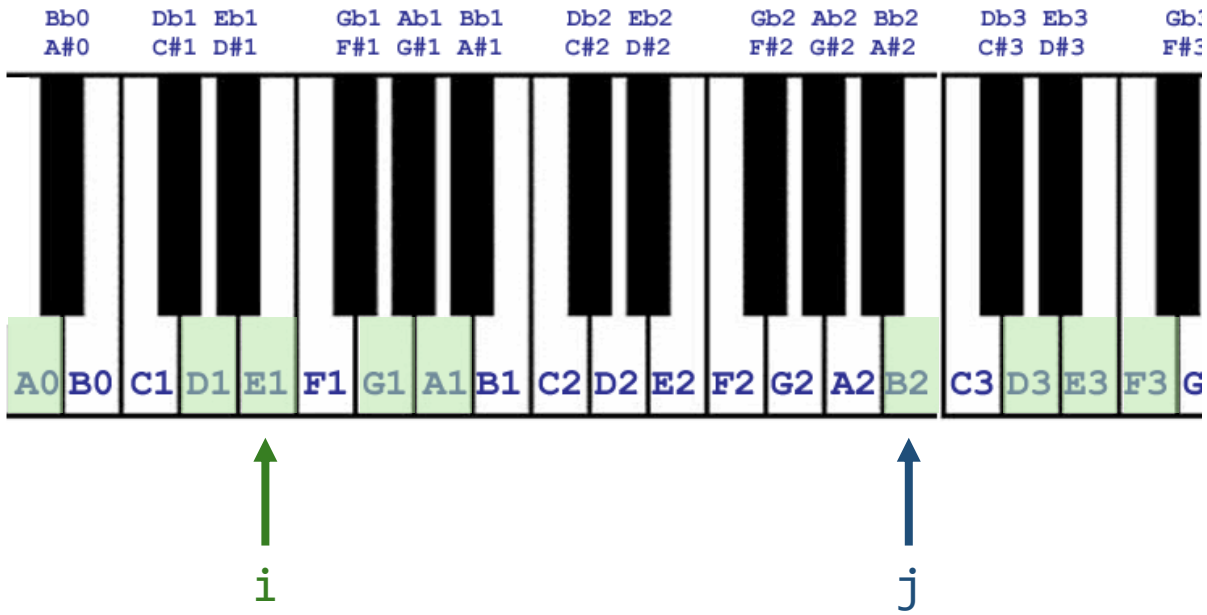
Input	9 8 A0 D1 E1 G1 A1 B2 D3 E3 F3



Outer loop	Position (i)	D1
	Furthest	A1
	Key pressed	4
Inner loop	Position (j)	F3
	Furthest	F3
	Key pressed	1
Sum of this combination		5
Current answer		8

Visualisation

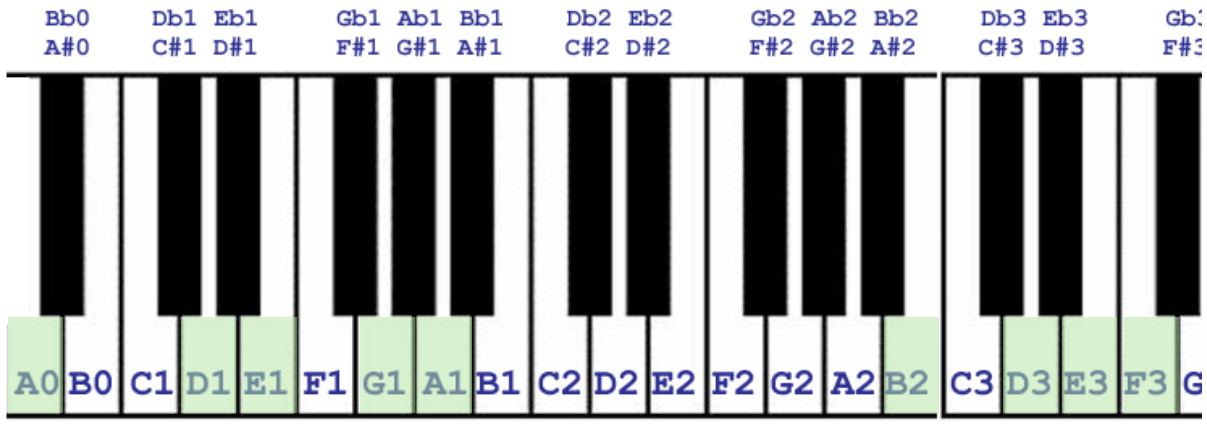
Input	9 8 A0 D1 E1 G1 A1 B2 D3 E3 F3



Outer loop	Position (i)	E1
	Furthest	A1
	Key pressed	3
Inner loop	Position (j)	B2
	Furthest	F3
	Key pressed	4
Sum of this combination		7
Current answer		8

Visualisation

Input	9 8 A0 D1 E1 G1 A1 B2 D3 E3 F3

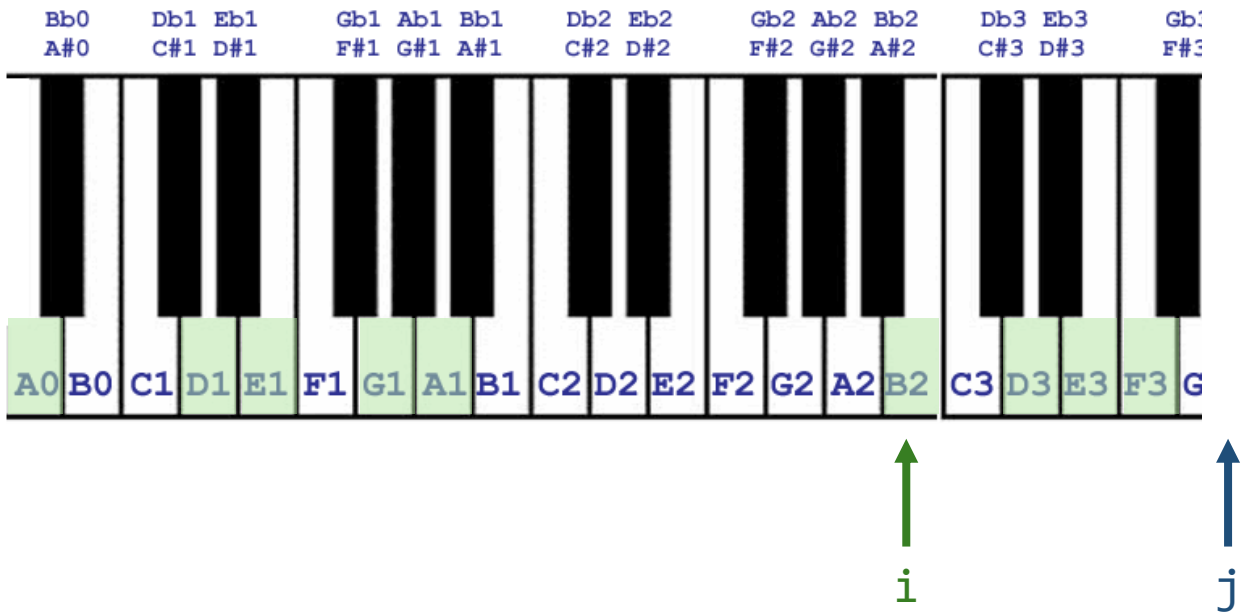


2000 years later…

Current answer	8

Visualisation

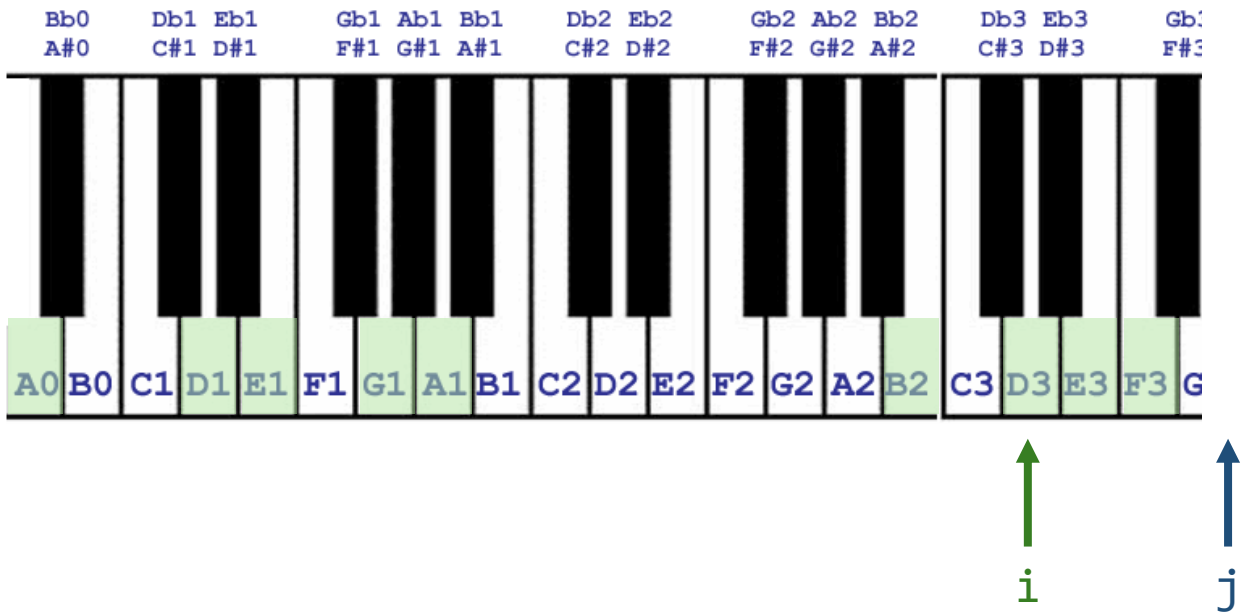
Input	9 8 A0 D1 E1 G1 A1 B2 D3 E3 F3



Outer loop	Position (i)	B2
	Furthest	F3
	Key pressed	4
Inner loop	Position (j)	-
	Furthest	-
	Key pressed	0
Sum of this combination		4
Current answer		8

Visualisation

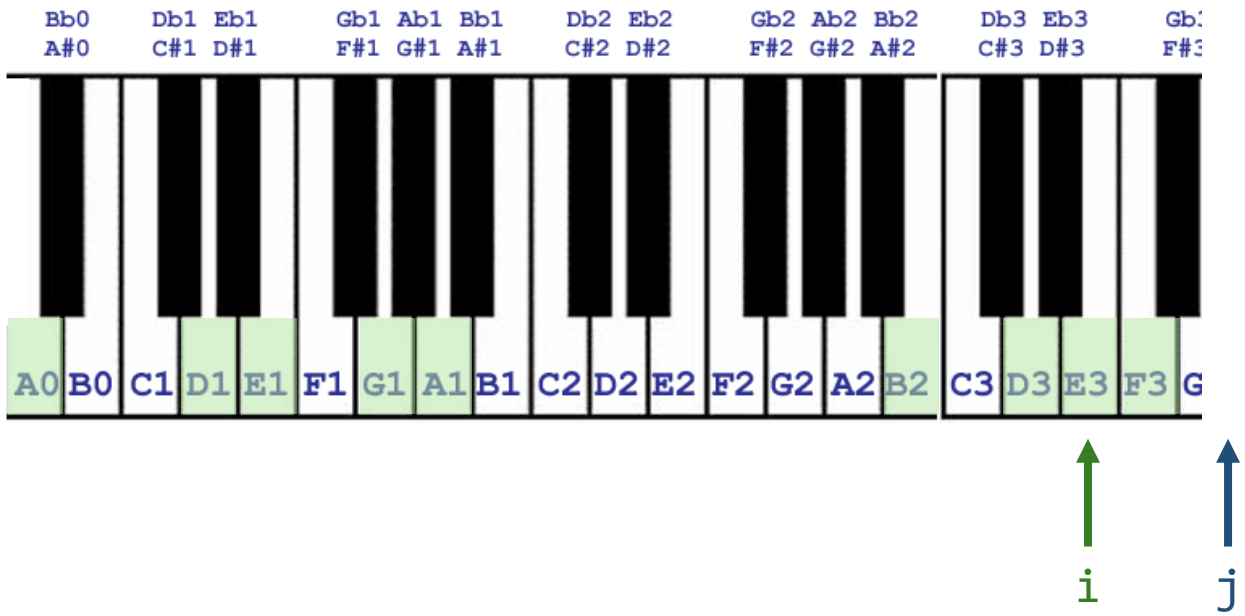
Input	9 8 A0 D1 E1 G1 A1 B2 D3 E3 F3



Outer loop	Position (i)	D3
	Furthest	F3
	Key pressed	3
Inner loop	Position (j)	-
	Furthest	-
	Key pressed	0
Sum of this combination		3
Current answer		8

Visualisation

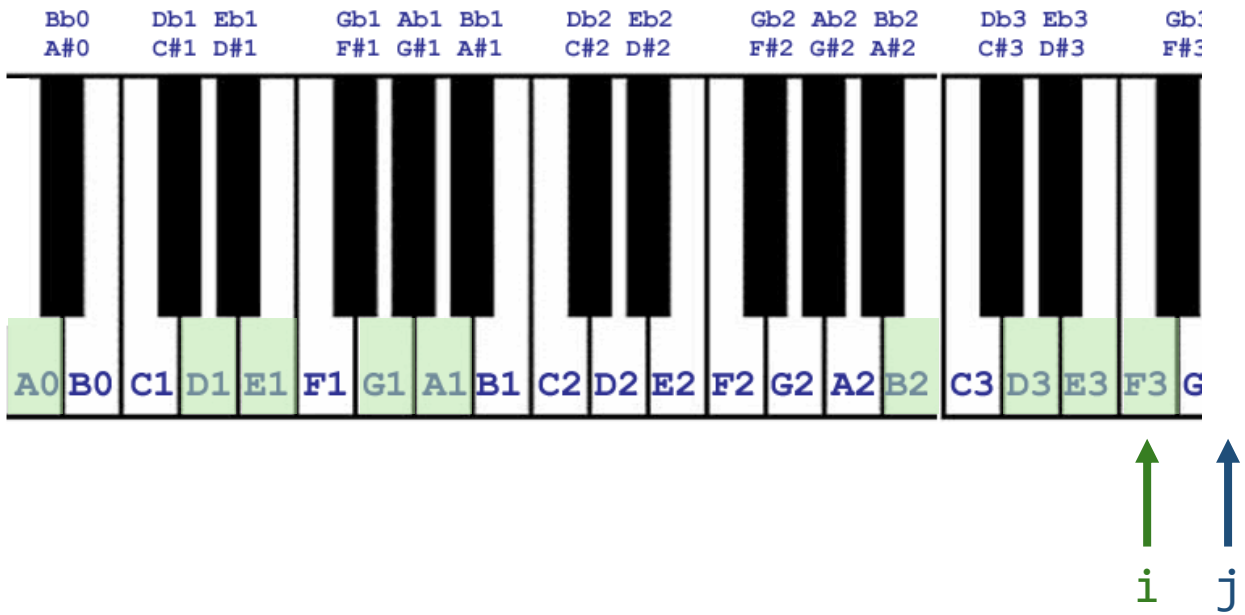
Input	9 8 A0 D1 E1 G1 A1 B2 D3 E3 F3



Outer loop	Position (i)	E3
	Furthest	F3
	Key pressed	2
Inner loop	Position (j)	-
	Furthest	-
	Key pressed	0
Sum of this combination		2
Current answer		8

Visualisation

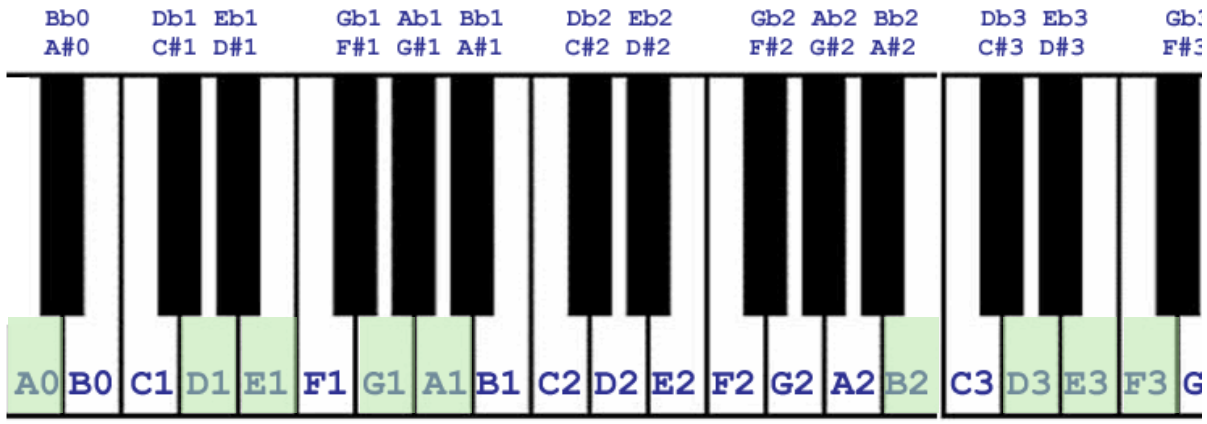
Input	9 8 A0 D1 E1 G1 A1 B2 D3 E3 F3



Outer loop	Position (i)	F3
	Furthest	F3
	Key pressed	1
Inner loop	Position (j)	-
	Furthest	-
	Key pressed	0
Sum of this combination		1
Current answer		8

Visualisation

Input	9 8 A0 D1 E1 G1 A1 B2 D3 E3 F3
Output	8



Outer loop	Position (i)	
	Furthest	
	Key pressed	
Inner loop	Position (j)	
	Furthest	
	Key pressed	
Sum of this combination		
Current answer		8

Subtask 2

$1 \leq N \leq 12$

All the notes are in octave 1

It maybe difficult to calculate the position of the keys and the distance between two keys.

You can just use some `if` statements to determine the position of the keys in this subtask.

Subtask 3

$1 \leq N \leq 52$

There is no # or b signs in the notes

Note that the black keys adopt the “one key, two names” policy.

For example, D♯ = E♭.

If you forget to handle this, you can still get this subtask.

Subtask 1

$1 \leq N \leq 7$

All the notes are in octave 1

There is no # or b signs in the notes

From subtask 2, you can just use some `if` statements to determine the position of the keys in this subtask.

From subtask 3, if you forget to handle "one key, two names", you can still get this subtask.

Hints

Use `map` and some Maths to find the position of the keys.

You may use integers (0-87 / 1-88 / whatever you like) to represent the positions.

Write the loops carefully, accessing out-of-bound data will cause you Runtime Error or Wrong Answer.

Check if you already go out-of-bound in every iteration.

`break` it immediately when out-of-bound happens.

There may exists better method than this, you may try thinking if there is any better way to tackle this problem.