awoo's blog

Educational Codeforces Round 114 Editorial

By **awoo**, history, 10 months ago, translation,

1574A - Regular Bracket Sequences

Idea: **BledDest**

Tutorial

1574A - Regular Bracket Sequences

There are many ways to solve this problem. The model solution does the following thing:

- start with the sequence `()()()()...`;
- merge the first 4 characters into one sequence to get `(())()()...`;
- merge the first 6 characters into one sequence to get `((()))()...`;
- and so on.

Solution (BledDest)

```
t = int(input())
for i in range(t):
    n = int(input())
    for j in range(n):
        print("()" * j + "(" * (n - j) + ")" * (n - j))
```

1574B - Combinatorics Homework

Idea: **BledDest**

Tutorial

1574B - Combinatorics Homework

Let's start with a simple assumption. For some fixed values a, b, c, the values of m that the answers exist for, make up a range. So there's the smallest possible number of adjacent equal pairs one can construct and the largest one — everything in-between exists as well.

The largest number is simple — put all A's, then all B's, then all C's. So this value is $(a-1)+(b-1)+(c-1)$.

The smallest number is trickier. Let's instead investigate when it's equal to 0. WLOG, assume $a \le b \le c$. Imagine the following construction. There are c letters C which separate blocks of letters A and B. There are $c-1$ ($c+1$ if you consider the ones to the sides of all letters C, but we want the smallest value, so we shouldn't consider them) such blocks, thus it's possible that each block contains no more than one letter A and no more than one letter B. So letters A and B will never produce adjacent pairs.

If there are empty blocks, then there are adjacent letters C. So the condition to still have no empty blocks is to have at least $c-1$ letters A and B in total. If $c-1 > a+b$, then any extra letter C can only be put adjacent to another letter C, thus producing an extra pair (at least one extra pair, but since we are examining the lower bound, we can always do exactly one). That means that the lower bound is $c-1-(a+b)$.

Now for the proof of the fact that every value in-between is also achievable. Since we have a construction for $m = 0$, let's try modifying it. Let's reduce the test to $m = 0$ the following way. While $m > 0$, decrease the count of the letter that appears the most by 1 and decrease m by 1. Now build the string for $m = 0$ with the reduced values. After that put the letters back, placing them next to the last occurrence of the same letter (there is at least one occurrence of each letter, the proof is trivial). That increases m by 1 and the count of this letter by 1. Thus, we'll return to the initial test.

Overall complexity: $O(1)$ per testcase.

Solution (awoo)

```
for _ in range(int(input())):
        a, b, c, m = map(int, input().split())
        a, b, c = sorted([a, b, c])
        print("YES" if c - (a + b + 1) <= m <= a + b + c - 3 else "NO")
```

1574C - Slay the Dragon

Idea: **BledDest**

Tutorial

1574C - Slay the Dragon

It is enough to consider two cases: whether we will increase the strength of the hero who will kill the dragon or not.

If you do not increase the hero's strength, then you have to choose such i that $a_i \geq x$. Obviously, among such i, you have to choose with the minimum value a_i, because the strength of defending heroes is equal to $\sum_{j=1}^{n} a_j - a_i$. It remains to increase the total strength of the remaining heroes to y. So the required number of coins is equal to $\max(0, y - (\sum_{j=1}^{n} a_j - a_i))$.

If you increase the hero's strength, then you have to choose the maximum value of a_i, which is less than x. In this case, the required number of coins is $x - a_i$ to increase the strength of the hero who will kill the dragon, plus $\max(0, y - (\sum_{j=1}^{n} a_j - a_i))$ to increase the strength of the defending heroes.

To find the heroes with strength as close to x as possible, you can use binary search (don't forget to sort the heroes beforehand).

Solution (Neon)

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

using namespace std;

using li = long long;

int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  int n;
  cin >> n;
  vector<li> a(n);
  for (auto &x : a) cin >> x;
  sort(a.begin(), a.end());
  li sum = accumulate(a.begin(), a.end(), 0LL);
  int m;
  cin >> m;
  while (m--) {
    li x, y;
```

→ Recent actions

Pranjallko → Why codeforces is not acceoting this code when all the outputs are mathing correctly?

anonymous112233 → HELP NEEDED: HARD RANGE QUERY PROBLEM

awoo → Educational Codeforces Round 132 Editorial

ScorpioDagger → Finally Cyan! UPD: Now, BLUE!

Vladosiya → Codeforces Round #811 (Div. 3)

WhyAsh5114 → ZCO eligibility for students choosing non-conventional but valid education (India)

AAK → Indian ICPC 2022 Regionals — Discussion

Cirno_9baka → CodeTON Round 2 Editorial

QAQAutoMaton → IOI2022 China Team

AquaMoon → CodeTON Round 2

Ghosted → Maybe Codeforces needs a more complete anti cheating system

Qingyu → XXII Open Cup, Grand Prix of China (EC-Finals)

chokudai → AtCoder Beginner Contest 262 Announcement

hemant_thakur → Help needed in DP problem

Yubai → An Interesting combination problem

jli505 → TeamsCode Summer 2022 Contest

Iftekhar_Hakim_K → Invitation to Replay of BUET Inter University Programming Contest 2022

marinyordanov → IATI 2018 (ban?) (Note that this is my new profile and therefore i am newbie :))

scipianus → Codeforces Round #271 (Div. 2) Editorial

SyadouHayami → Ask for the solution of CF1704F

abhinav700 → problem while declaring a 3d array

dsbdsb → fighting for rating 2200

RDFZzzx → A brief introduction of Segment Tree(I):Build tree and query without update

1900_mashups → Mashups for Aspiring Masters.

MikeMirzayanov → Rule about third-party code is changing

Detailed →

```
    cin >> x >> y;
    int i = lower_bound(a.begin(), a.end(), x) - a.begin();
    li ans = 2e18;
    if (i > 0) ans = min(ans, (x - a[i - 1]) + max(0LL, y - sum + a[i - 1]));
    if (i < n) ans = min(ans, max(0LL, y - sum + a[i]));
    cout << ans << '\n';
  }
}
```

1574D - The Strongest Build

Idea: **BledDest**

Tutorial

1574D - The Strongest Build

Consider the bruteforce solution. You start with a build that contains the most powerful item for each slot. In one move, you swap an item in some slot for the one that is the previous one by power. If a build is not banned, update the answer with its total power (banned builds can be stored in a set, maybe hashset if you hash carefully enough).

Notice that if you reach some unbanned build in this bruteforce, it never makes sense to go further. The answer is already updated with this one, and all the lower ones have smaller power.

If you code that bruteforce in a smart way (or just add memorization), you won't visit any build twice. How many states will you visit, though? Since you can only proceed if you are standing in a banned build, you will check around $m + mn$ builds.

You can code it like that and get accepted. However, there's another way that's easier to code, in my opinion.

The optimal answer can be one of only two types. Either it contains the last item of each slot. Or it's some banned build with one item swapped with the previous one. It's easy to see from the solution above. So you can check the first type, then iterate over the banned build and try swapping each slot in it, checking if the resulting build is banned or not.

Overall complexity: $O(mn)$ or $O(mn \log m)$.

Solution (awoo)

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

#define forn(i, n) for (int i = 0; i < int(n); i++)

using namespace std;

int n;
vector<vector<int>> a;
int m;
vector<vector<int>> b;

int main() {
        scanf("%d", &n);
        a.resize(n);
        forn(i, n){
                int c;
                scanf("%d", &c);
                a[i].resize(c);
                forn(j, c) scanf("%d", &a[i][j]);
        }
        scanf("%d", &m);
        b.resize(m);
        forn(i, m){
                b[i].resize(n);
                forn(j, n){
```

```
                        scanf("%d", &b[i][j]);
                        --b[i][j];
                }
        }
        sort(b.begin(), b.end());
        vector<int> ult(n);
        forn(i, n) ult[i] = int(a[i].size()) - 1;
        if (!binary_search(b.begin(), b.end(), ult)){
                forn(i, n) printf("%d ", ult[i] + 1);
                puts("");
                return 0;
        }
        int mx = 0;
        vector<int> ans(n, -1);
        forn(i, m){
                vector<int> tmp = b[i];
                int sum = 0;
                forn(j, n) sum += a[j][tmp[j]];
                forn(j, n) if (tmp[j] != 0){
                        --tmp[j];
                        if (!binary_search(b.begin(), b.end(), tmp) && sum -
a[j][tmp[j] + 1] + a[j][tmp[j]] > mx){
                                mx = sum - a[j][tmp[j] + 1] + a[j][tmp[j]];
                                ans = tmp;
                        }
                        ++tmp[j];
                }
        }
        forn(i, n){
                printf("%d ", ans[i] + 1);
        }
        puts("");
}
```

1574E - Coloring

Idea: **Roms**

Tutorial

1574E - Coloring

For best understanding we replace the matrix with 0 and 1 with the matrix with black and white cells.

At first let's consider matrix if there are two adjacent horizontal cell with same color (for example cells $(5, 5)$ and $(5, 6)$ are black). Then the cells $(4, 5)$, $(4, 6)$, $(6, 5)$ and $6, 6$ must have the opposite color (white); the cells $(3, 5)$, $(3, 6)$, $(7, 5)$ and $7, 6$ must have the same color (black) and so on. So, two adjacent horizontal cells generate the *vertical strip* of width two. Reciprocally two adjacent vertical cells generate the *horizontal strip* of width two. And if simultaneously there are *horizontal strip* and *vertical strip* then the answer is 0 (because they contradict each other).

If there are two cells of same color in the same row with even number of cells between them (for example $(2, 2)$ and $(2, 7)$ with four cells between them) then there is the *vertical strip* (because there are always two adjacent cells with same color between them). The same is correct for *horizontal strips*.

Now let's consider how the matrix look if there are the *vertical strip*. It look like a chess board of size $n \times m$, but colors of some verticals are inverted. The same is correct if there are the *horizontal strips*.

How we can quickly understand that there are two cells of same color in the same row with even number of cells between them? For this mentally color the matrix in a checkerboard pattern. And then one of this cells has the same color witch cells in chessboard, and the other has the opposite color witch cells in chessboard.

For calculating the answer we have maintain to the following values:

- The color of each colored cell;
- The row and columns containing the cells of same color with even number of cells between them;
- And the number of row and columns containing at least one colored cell (for calculating the number of beautiful matrix).

Solution (Roms)

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

using namespace std;

const int MOD = 998244353;
const int N = 1'000'009;

int sum (int a, int b) {
    int res = a + b;
    if (res < 0) res += MOD;
    if (res >= MOD) res -= MOD;
    return res;
}

int n, m, k;
map <pair <int, int>, char> c;
int cntr[N][2], cntc[N][2];
int cntx[2];
set <int> badr, badc;
set<int> ur, uc;
int p2[N];

void upd2(int pos, int col, int add, int cnt[N][2], set <int> &bad, set<int>
&u) {
    cnt[pos][col] += add;
    assert(cnt[pos][col] >= 0);

    if (cnt[pos][0] > 0 && cnt[pos][1] > 0){
        if (!bad.count(pos))
            bad.insert(pos);
    } else {
        if (bad.count(pos))
            bad.erase(pos);
    }

    if (cnt[pos][0] > 0 || cnt[pos][1] > 0){
        if (!u.count(pos))
            u.insert(pos);
    } else {
        if (u.count(pos))
            u.erase(pos);
    }
}

void upd(int x, int y, int t) {
    int col = (x & 1) ^ (y & 1);
    if (c.count({x, y})) {
        int ncol = col ^ c[{x, y}];
        --cntx[ncol];
        upd2(x, ncol, -1, cntr, badr, ur);
        upd2(y, ncol, -1, cntc, badc, uc);
        c.erase({x, y});
    }

    if (t == -1)
        return;
```

```
    int ncol = col ^ t;
    ++cntx[ncol];
    upd2(x, ncol, 1, cntr, badr, ur);
    upd2(y, ncol, 1, cntc, badc, uc);
    c[{x, y}] = t;
}

int main(){
    p2[0] = 1;
    for (int i = 1; i < N; ++i)
        p2[i] = sum(p2[i - 1], p2[i - 1]);

    scanf("%d%d%d", &n, &m, &k);
    for (int i = 0; i < k; ++i) {
        int x, y, t;
        scanf("%d %d %d", &x, &y, &t);
        --x, --y;
        upd(x, y, t);

        int res = 0;
        if(badr.size() > 0 && badc.size() > 0) {
            res = 0;
        } else if (badr.size() > 0) {
            assert(m - uc.size() >= 0);
            res = p2[m - uc.size()];
        } else if (badc.size() > 0) {
            assert(n - ur.size() >= 0);
            res = p2[n - ur.size()];
        } else {
            if (ur.size() == 0 && uc.size() == 0)
                res = sum(sum(p2[n], p2[m]), -2);
            else {
                assert(m - uc.size() >= 0);
                res = sum(res, p2[m - uc.size()]);
                assert(n - ur.size() >= 0);
                res = sum(res, p2[n - ur.size()]);
                if (cntx[0] == 0 || cntx[1] == 0) {
                    assert(cntx[0] != 0 || cntx[1] != 0);
                    res = sum(res, -1);
                }
            }
        }

        printf("%d\n", res);
    }
    return 0;
}
```

1574F - Occurrences

Idea: **BledDest**

Tutorial

1574F - Occurrences

What does the condition "the number of occurrences of A_i in the array a is **not less** than the number of occurrences of each non-empty subarray of A_i in a" mean? First, if A_i contains two (or more) equal elements, then any occurrence of A_i introduces at least two occurrences of that element; so any element in A_i is forbidden (it should not appear in the resulting array). Now let's consider an array A_i such that every its element is unique. Every element of A_i should be a part of an occurrence of A_i in the array a. Let's rephrase this condition as follows: **for each occurrence of $A_{i,j}$ in a, the next element in a is $A_{i,j+1}$, and vice versa: for each occurrence of $A_{i,j+1}$ in a, the previous element in a is $A_{i,j}$.**

Let's build a directed graph on k vertices, where an arc from vertex x to vertex y means that each occurrence of x should be followed by y, and each occurrence of y should be preceded by x (i. e. x is followed by y in some array A_i). Let's consider the weakly connected components in this graph. If we have at least one occurrence of some element from a component in a, it means that all other elements from this component occur in a as well. Some integers from 1 and k are "bad" in a sense that we cannot uniquely determine which element should follow/precede them (in terms of graph theory, it means that the in-degree or out-degree of a vertex is at least 2). Since by picking one element from a component, we will have to use all elements from a component, it means that if a component contains at least one "bad" element, the whole component will be "bad" — we cannot use any element from it.

If a component is a cycle, no vertex has in-degree or out-degree greater than 1, but the component is still "bad" since, if we include at least one element from a, we cannot finish the cycle — the array a is not infinite, but the cycle is.

Okay, the only "good" components are chains. When we use an element from a chain in a, all elements from this chain will be used in exactly the same order that they were in the chain; so, a should consist of some chains linked together (chains may repeat, and some chains may be absent from a). We can write a solution with dynamic programming: let dp_i be the number of ways to construct an array of length i using these chains. The transitions are as follows: $dp_i = \sum\limits_{j=1}^{c} dp_{i-len_j}$, where c is the number of chains, and len_j is the length of the j-th chain.

The number of chains is up to k, and the number of states in dynamic programming is $m + 1$, so the solution works in $O(mk)$, which is too slow. We can improve it with the following two facts:

- all chains of the same length are indistinguishable;
- there are $O(\sqrt{k})$ different lengths of chains.

So, instead of iterating on the chains themselves in dynamic programming, we will iterate on the lengths of the chains (considering only lengths having at least one chain), and process all chains of the same length as one by introducing a multiplier in our dynamic programming: $dp_i = \sum\limits_{j=1}^{k} dp_{i-j} \cdot cnt_j$, where cnt_j is the number of chains of length j. That way, our dynamic programming will work in $O(m\sqrt{k})$ if we skip the values of j with $cnt_j = 0$.

Solution (BledDest)

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

using namespace std;

const int MOD = 998244353;

int add(int x, int y)
{
        x += y;
        while(x >= MOD) x -= MOD;
        while(x < 0) x += MOD;
        return x;
}

int mul(int x, int y)
{
        return (x * 1ll * y) % MOD;
}

int main()
{
        int n, m, k;
        scanf("%d %d %d", &n, &m, &k);
        vector<vector<int>> A(n);
        vector<int> bad_num(k);
```

```
        for(int i = 0; i < n; i++)
        {
                int c;
                scanf("%d", &c);
                A[i].resize(c);
                for(int j = 0; j < c; j++)
                {
                        scanf("%d", &A[i][j]);
                        A[i][j]--;
                }
        }

        for(int i = 0; i < n; i++)
        {
                if(set<int>(A[i].begin(), A[i].end()).size() != A[i].size())
                {
                        for(auto x : A[i])
                                bad_num[x] = 1;
                }
        }
        vector<vector<int>> nxt(k);
        vector<vector<int>> prv(k);
        for(int i = 0; i < n; i++)
                for(int j = 0; j + 1 < A[i].size(); j++)
                {
                        nxt[A[i][j]].push_back(A[i][j + 1]);
                        prv[A[i][j + 1]].push_back(A[i][j]);
                }

        for(int i = 0; i < k; i++)
        {
            sort(nxt[i].begin(), nxt[i].end());
            sort(prv[i].begin(), prv[i].end());
                nxt[i].erase(unique(nxt[i].begin(), nxt[i].end()),
nxt[i].end());
                prv[i].erase(unique(prv[i].begin(), prv[i].end()),
prv[i].end());
        }

        vector<int> used(k, 0);
        vector<int> cnt(k + 1, 0);
        for(int i = 0; i < k; i++)
        {
                if(used[i]) continue;
                queue<int> q;
                vector<int> comp;
                q.push(i);
                used[i] = 1;
                while(!q.empty())
                {
                        int z = q.front();
                        q.pop();
                        comp.push_back(z);
                        for(auto x : nxt[z])
                                if(!used[x])
                                {
                                        used[x] = 1;
                                        q.push(x);
                                }
                        for(auto x : prv[z])
                                if(!used[x])
                                {
                                        used[x] = 1;
                                        q.push(x);
                                }
```

```
                }
                bool bad = false;
                int cnt_beg = 0;
                for(auto x : comp)
                {
                        if(prv[x].empty())
                                cnt_beg++;
                        if(prv[x].size() > 1 || nxt[x].size() > 1 ||
bad_num[x])
                                bad = true;
                }
                bad |= (cnt_beg == 0);
                if(!bad)
                        cnt[comp.size()]++;
        }
        vector<int> nonzero;
        for(int i = 1; i <= k; i++)
                if(cnt[i] > 0)
                        nonzero.push_back(i);
        vector<int> dp(m + 1, 0);
        dp[0] = 1;
        for(int i = 1; i <= m; i++)
                for(auto x : nonzero)
                        if(x <= i)
                                dp[i] = add(dp[i], mul(cnt[x], dp[i - x]));

        printf("%d\n", dp[m]);
        return 0;
}
```

Tutorial of Educational Codeforces Round 114 (Rated for Div. 2)

+122 awoo 10 months ago 90

Comments (90)

Write comment?

AHF

10 months ago, # | 0

Thanks :)

→ Reply

onglogn

10 months ago, # | 0

nice editorial, love the problems, appreciate it

→ Reply

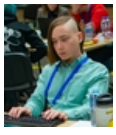
Maurycyt

10 months ago, # | 0

In problem E, we only have to keep track of same-colour cells with an even number of cells between them?

What about differently coloured cells with an odd number of cells between them? Doesn't that force a strip of width two as well?

→ Reply



maxplus

10 months ago, # ^ | +1

Same colour is just a special case like adjacent cells, mentioned to arrive to the checkerboard intuition which works both for same colour and opposite colour cases without distinguishing them.

→ Reply

10 months ago, # ^ | 0

@maxplus can you explain Problem E. I am new to CP. Like

micky2100

@maxplus can you explain Problem E. I am how to C . Like for input 2 2 7 1 1 1 1 2 1 2 1 1 1 1 0 1 2 -1 2 1 -1 1 1 -1

o/p: 3 1 0 1 2 3 6 why 1 1 1 gives 3 as o/p . Initially the matrix is empty so there are 2 ways to fill it, either 0 or 1. then the answer should be 2 right. Similarly for 2 1 -1 whey o/p is 3. 2,1 wasn't filled yet, so it can filled in 2 ways with 0 or 1.

→ Reply

Maurycyt

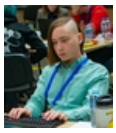
10 months ago, # ^ | ☆ 0

When the 2×2 matrix is empty, there are actually 6 ways to fill it such that every square will add to 2. Here are the options:

Options

So when we set the cell with coordinates $(1, 1)$ we get only first 3 options out of the above.

→ Reply



maxplus

10 months ago, # ^ | ☆ +8

I guess `* The row and columns containing the cells of same color with even number of cells between them;` can be confusing, so you're right, it's lines containing cells that correspond to both (contradicting) checkerboard line colourings that should be tracked.

→ Reply

Maurycyt

10 months ago, # ^ | ☆ 0

Thanks!

→ Reply

Maurycyt

10 months ago, # ^ | ☆ +5

OK, I implemented my own version and I think the code is a bit more self-explanatory. If anyone has trouble understanding Roms' implementation (as I did), then maybe looking at mine (129556708) will help.

→ Reply

wargang

10 months ago, # | ☆ ← Rev. 3 -44

→ Reply

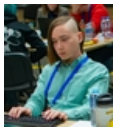
The comment is hidden because of too negative feedback, click here to view it

agarus

10 months ago, # | ☆ 0

Problem E. What is cntx[] in Roms's solution and why are we (--res) only if it [0,1] or [1,0]. Shouldn't we always (--res) because same starting position shared between p2[n-ur] and p2[m-uc]?

→ Reply



maxplus

10 months ago, # ^ | ☆ +3

cntx to the whole board as cntr to a row. If there is at least one strip, there are no colourings alternating nicely both vertically and horizontally so none are subtracted, and if no cells are coloured then there are two "shared" colourings.

→ Reply

agarus

10 months ago, # ^ | ☆ 0

Thank you. Got it: cntr and cntc keep parity info for each col and row. While cntx keep parity for all board — how many of them correspond to their (virtual) color and how many doesn't. But what information does it provide us — why should we (--res) in case all {correspond} or all {notcorrespond}? It tells us if there can be shared state in the start — the position from

which we can shift either vertical lines or horizontal. If there

points in both {correspond} and {notcorrespond} positions — there cannot be such common starting position.

→ Reply


Iterativ

10 months ago, # | ☆ 0

Is there a way of solving problem D with a trie?

→ Reply


LetterC67

10 months ago, # ^ | ☆ ← Rev. 3 +16

Yes, I solved D with Trie and DP (unfortunately I couldn't complete it during the contest).

Here is the code

The idea is put all the banned set into a trie. Then iterate through every possible prefixes. For each prefix, we will find the maximum possible suffix so that they will not form a banned set.

For example we have a prefix of length i, there are 6 items for slot $i+1$, current node has 3 children numbered 3, 5, 6. So we may choose $4th$ item for slot $i+1$ and choose whatever we want from slot $i+2$

The idea is simple but one should be careful while implement

→ Reply

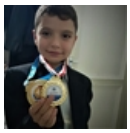

Maurycyt

10 months ago, # ^ | ☆ +1

As far as I can tell, I think I did it using a traverse of a trie structure without the structure itself. Take a look at my solution if you want to.

129399828

→ Reply


AmmarDab3an

10 months ago, # ^ | ☆ +3

you can check my implementation

→ Reply


pallav12

10 months ago, # | ☆ 0

I saw everyone using PriorityQueue for D, can anyone explain that approach.

→ Reply


Maurycyt

10 months ago, # ^ | ☆ +5

People used a funky priority_queue based Dijkstra implementation! A very clever approach in my opinion.

Each vertex is a build and each build is a vertex. However, these vertices aren't stored anywhere in memory but rather, they are procedurally generated. Two vertices (builds) are joined by an edge if and only if you get one of the builds by degrading exactly one item in the other build.

Then, just implement a Dijkstra on these builds and remember to skip the neighbours of a vertex if the vertex is not banned, since the neighbours (degraded builds) can only be worse, as noted in the editorial.

→ Reply


xiaohaixu

10 months ago, # ^ | ☆ +8

The idea is basically to store the build in the priority queue, and we use custom comparator so that builds in PQ are sorted according to their total strength. We add the strongest build first to the PQ (in this case, the last item in each slot). When PQ is not empty, check if its top is not banned, if so then we get the answer. Otherwise we iteratively swap an item in some slot for the one that is the previous one by power and add

to PQ.

→ Reply

pallav12

10 months ago, # ^ | ☆ 0

Got it... very clean.

→ Reply

am_chourasia

10 months ago, # ^ | ☆ 0

Thanks, I saw your solution and I must say it was very clear...

→ Reply

ganesh_6

10 months ago, # ^ | ☆ 0

very well articulated

→ Reply

ganesh_6

10 months ago, # ^ | ☆ 0

I have implemented the same algorithm as yours but i am getting TLE in java. Any suggestions for better complexity? I did a similar one using dfs in java, it worked. But funky dijkstras based one(same as yours) is getting TLE on T15. Please check my submission: https://codeforces.com/contest/1574/submission/129776871

→ Reply

abhiaps

10 months ago, # | ☆ 0

In D part,i did simple brute force approach with some sort of memorization. But to my extreme surprise the code got accepted.

129377465

By the way one of the best contest for me. This contest boosted my rating to a great extent.

→ Reply

vaal

10 months ago, # ^ | ☆ ← Rev. 2 0

lol its giving TLE for me though. My submission : https://codeforces.com/contest/1574/submission/129585401. Update : yeah i got it why im getting TLE

→ Reply

Relief

10 months ago, # ^ | ☆ 0

What was the reason for TLE in your case?

→ Reply

vaal

10 months ago, # ^ | ☆ ← Rev. 2 0

for (; max[ind]>=0; --max[ind]) { solve(ind+1, max, pos, set); } this loop makes the worst case complexity to O(10^(5n)).

→ Reply

Relief

10 months ago, # ^ | ☆ 0

I'm also getting TLE #6, but I haven't found the issue. Here's my code

→ Reply

10 months ago, # ^ | ☆ 0

Oh, sorry man, I cant understand the operations which

u have made since I dont know

vaal

Java that much!!

→ Reply

singleLOOP

10 months ago, # | ☆ 0

great short approach for D

→ Reply

akulenok

10 months ago, # | ☆ 0

Can somebody please explain why we do

```
if (ur.size() == 0 && uc.size() == 0)
    res = sum(sum(p2[n], p2[m]), -2);
```

and

```
if (cntx[0] == 0 || cntx[1] == 0)
    res = sum(res, -1);
```

in E? Thanks in advance.

→ Reply

Maurycyt

10 months ago, # ^ | ☆ ← Rev. 2 +3

I do not understand what the variable names mean exactly but based on what I've written in my solution, the first bit of code should be the embodiment of the following train of thought:

```
if there are no forced stripes (horizontal or vertical) and
no rows or columns have any cell coloured in them (they are
not set in place) then
the possible number of colourings is:
Assume there are horizontal stripes, that gives 2^n
different colourings.
Assume there are vertical stripes, that gives 2^m different
colourings.
But the two cases above share two colourings, which is the
checkerboard pattern and the anti-checkerboard pattern,
which have to be subtracted from the total.
```

The second bit of code I couldn't decipher, but it's likely something similar.

→ Reply

agarus

10 months ago, # ^ | ☆ ← Rev. 4 +3

ur == used in rows; uc == used in columns. It keeps indexes of rows and columns that already used — ones that have 1's and 0's written in one of their cell.

Now every possible permutation can be achieved in one of two ways: either by left-right shift/move of horizontal lines or by up-down shift/move of vertical line. And every shift is creating doubled-lines in that direction — except starting position end ending. They are identical (if we consider starting position — a permutation that looks like a chessboard, and ending position — 2^n or 2^m positions, when all rows or all columns are shifted (inverted chessboard)). This means that every vertical shift creates position unachievable by horizontal shift and vice versa. Except starting and ending positions — if we shift non vert (or all vert) — thay will be the same like if we would shift non (or all) horizontal. If both of ur and uc them is empty — we haven't written any numbers — both starting and ending positions are achievable, so we should substruct 2. If there at least one cell with written 1 or 0 — we wouldn't have same starting and ending position. We could maximum have only one of those 2 positions — corresponding to written number and its cell.

cntx is variable that keeps count of <black & white> 'parity'. If you'll

cntx is variable that keeps count of <black & white> parity. If you'll imagine an m*n chess board with black square in upper left corner (that is {1,1} coordinate), than you can say that cntx[0] is counting how many 0s you placed on black cell and 1s on whites. cntx[1] is counting how many 0s you placed on white and 1s on black. If there some 1s (or 0s) on both black and whites — there cannot be same starting and ending position. We should not decrease result by on. But if all 1's are on same color and all 0s on other — there could be match for one position — this position is both contained in sum(res, p2[m — uc.size()]) and sum(res, p2[n — ur.size()]), so we should substruct it.

→ Reply


Keep.It.Simple

10 months ago, # ^ | ☆ 0

Sorry for asking noob question. I still don't understand why the total permutation is 2^n+2^m-2 (like the way to count up to a total of 2^n+2^m, then subtract 2 cases that is identical). Could you please explain in more detail with example.

Now every possible permutation can be achieved in one of two ways: either by left-right shift/move of horizontal lines or by up-down shift/move of vertical line. And every shift is creating doubled-lines in that direction

Thank you.

→ Reply


agarus

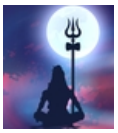
10 months ago, # ^ | ☆ 0

I don't know what to add. If you imagine a chess board of m*n — any permutation is achievable either by moving some horizontal "zebra-lines" 1 square left (or right — it doesn't matter, they are indistinguishable) OR by moving some vertical "zebra-lines" 1 square up (or down — doesn't matter). There 2^n variants of horizontal shifts — each line either shifted or not. And 2^m verticals — same logic.

If you observe 2^n horizontal shifting — there 1 "starting" position when none is shifted, and one "ending" position — when all shifted and chess board becomes inverted — they correspond to "starting" and "ending" position in 2^m vertical shifts. So those 2 should be substructed.

Case with cntx and -1 is much more interesting :)

→ Reply


Yaduvans_hi

10 months ago, # | ☆ 0

Problem D please help! Can someone explain me this statement, **"The optimal answer can be one of only two types. Either it contains the last item of each slot. Or it's some banned build with one item swapped with the previous one.**

I couldn't understand the 2nd part how the answer can be part of banned build with one swapped with previous one ?? Thanks in advance

→ Reply


Amimoy12

10 months ago, # ^ | ☆ 0

Suppose (b1,b2,b3,...bs) is a banned build.Then one of our possible candidate for the best possible build is (b1,b2,...bi-1,...bs), given bi>1.This build is adjacent to one of our banned build so if not banned too will be a potential candidate.

→ Reply


JvThunder

10 months ago, # | ☆ ← Rev. 6 +9

I actually used BFS to solve problem D. This might not be the fastest solution to implement, but I think it's easy to understand. Here's my idea.

The state of the BFS is the array b, which is the index of items chosen on each

The state of the BFS is the array b, which is the index of items chosen on each slot. The value of the state is the sum of the chosen items. We start with all index maxed out for each slot and manually calculate the sum of all the last index as the value. Every time we transition to another state, we decrease one of the $b[i]$ by 1, iterating i from 1 to n. Thus, the value would only be affected by one of the slot. To maintain the value, we need to remove the previous value and add the current value, i.e. the value would decrease by $a[i][b[i]] - a[i][b[i]-1]$ ($b[i]$ here is before we decrease by 1). By using BFS, we will get the states in descending order without skipping any states. We should use max heap for the BFS because we prioritize the higher sum. We would stop when the current state is not banned and output it as the answer.

Complexity Analysis: Since we would directly stop when it is not banned, the number of states visited would only be $O(M)$ at most. Since we use a priority queue and assuming we check a visited state using set/map, the complexity would be $O(N^2MlogNM)$.

My code: https://codeforces.com/contest/1574/submission/129387703 (Note that I used a slightly different value definition, which is the difference of the current sum with the maximum sum, but the main idea is still the same)

Edit: I miscalculated the complexity. Big thanks to **maxplus** for pointing this out.
→ Reply


MoHitman

10 months ago, # ^ | ☆ +5

Thank you so much, I was having the same approach but I was struggling very much implementing , you makes it looks so simple
→ Reply


JvThunder

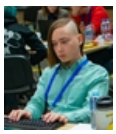
10 months ago, # ^ | ☆ +3

Glad to help!
→ Reply


JvThunder

10 months ago, # ^ | ☆ +3

Sorry just wanna ask, I'm wondering why "Glad to help!" got so many downvotes. Is it insulting or negative in any way?
→ Reply


maxplus

10 months ago, # ^ | ☆ ← Rev. 2 +13

Isn't it $O(N^2MlogNM)$? Although at most $M+1$ states are extracted from priority queue, for each of them $O(N)$ children are processed in $O(NlogNM)$ each.
→ Reply


JvThunder

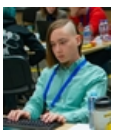
10 months ago, # ^ | ☆ 0

I see, my bad I think you're right, sorry I will fix that.
→ Reply


JvThunder

10 months ago, # ^ | ☆ 0

I am thinking perhaps one could improve this solution's complexity by using hashing perhaps? Although in this problem, the N is small so it still passes the time limit.
→ Reply


maxplus

10 months ago, # ^ | ☆ +5

With hashing we'll process each child in $O(N)$ since worst case we need to create each of them, so $O(N^2M + NMlogNM)$ (in your solution you'll need to provide pq with a custom comparator so that it can ignore state). N^2M is hard to shake off because it's also the space complexity independent of lookup algorithm. Instead of hashing we could visit children in such a way that each gets

exactly one parent — we can stop generating

exactly one parent — we can stop generating children after we encounter the first not maxed slot, same complexity.

If we use hashing, replace priority queue with queue, update best answer instead of terminating on the first not banned state and skip states whose children can't improve the answer, $O(N^2M)$.

If we update hash in $O(1)$, update current best answer before placing a child into the queue and skip useless children (that aren't better than current best answer), we get $O(NM)$ but it becomes hard not to notice that only banned states are placed into the queue and we don't need queue at all.

One also has to be careful with updating the best state if he wants to avoid N^2 (in the solution from the editorial too): if we assign it each time it gets improved, it can be up to NM updates of length-N array.

→ Reply

JvThunder

10 months ago, # ^ | ← Rev. 2 0

Nice! Thanks for the insight. I learn a lot even when my code have already got Accepted.

→ Reply

poiug07

10 months ago, # | ☆ 0

B. Why this code in GO is dealing reversed results when sent? On my local machine everything is working fine.

Spoiler

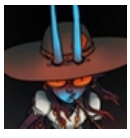
→ Reply

rhono_1

10 months ago, # | ☆ 0

Will anyone help me in upsolving C problem. I find that the editorial is giving TLE. Please help me

→ Reply



agarus

10 months ago, # ^ | ☆ ← Rev. 3 0

You may notice that test#5 has n = 2*10e5 took 1,7 secs. Test #6 has n = 2*10^5 size too (thought we don't know how big is m in both), but we can see that "a" now is not a bunch of 1s and 2s {1 2 1 2 1 2 1 2 ...} but big numbers like 873579811631 — which means they will took your algorithm longer to digest.

Take a look at "fast input output c++" topic. Google it.

And be sure to take a look at this.

→ Reply

Ankush08

8 months ago, # ^ | ☆ 0

Thanks, agarus, I was facing the same issue, and this helped me :)

→ Reply

Dzksh

10 months ago, # | ☆ 0

Even though I have used binary search to find the heroes in question **1574C — Slay the Dragon** it is giving TLE. Pls can someone check why is it giving so? my code is : 129554543

→ Reply

10 months ago, # ^ | ☆ ← Rev. 2 +3

Your input/output is too slow. This is caused by using iostream


Maurycyt

Your input/output is too slow. This is caused by using iostream (cin/cout) with stdio (printf/scanf) synchronisation. Add the following lines to boost the speed considerably.

```
ios_base::sync_with_stdio(0);
cin.tie(0);
```

Beware, however, that this makes it dangerous to use cin/cout alongside printf/scanf.

I submitted your code with these two lines and it works within the time limit (129564251), albeit still a bit slow.

→ Reply


Dzksh

10 months ago, # ^ | ☆ 0

Thanks a lot for helping me!

→ Reply


mahestippi

7 months ago, # ^ | ☆ 0

I don't know why(still) my code is getting TLE, but after writing those two lines it got accepted. Thanks for that reply. Here is my code which got me TLE CODE

→ Reply


C-3PO

10 months ago, # | ☆ 0

Hey, can someone please explain Geothermal's approach to question D:
https://codeforces.com/contest/1574/submission/129399246

It runs in approx 400 MS and my pq method is running in ~2900 MS:
https://codeforces.com/contest/1574/submission/129568774

→ Reply


LetterC67

10 months ago, # ^ | ☆ 0

A banned build is hashed into a pair in Geothermal's code. Your code just push the whole build into the heap, obviously it's much slower.

→ Reply


C-3PO

10 months ago, # ^ | ☆ 0

https://codeforces.com/submissions/Ashishgup# see this, he simply uses set

→ Reply


agarus

10 months ago, # ^ | ☆ 0

Yes, but he is not moving them through PQ.

What is impressive — is his strange "recurse" function with quick return shortcuts. And strange line < ans &= (i + 1 == c[idx]); >

→ Reply


C-3PO

10 months ago, # ^ | ☆ 0

yes :'), did you get his approach?

→ Reply


agarus

10 months ago, # ^ | ☆ 0

Yes, he is doing kind-of-dfs until he finds allowed build where all except current node are ones. Than he goes one node up and trying do the same. He is looking into same node that stupid-bfs does, but in different order. Thought bfs has chance of early

exit, so it might not visit every

blocked build, and this approach does not.

→ Reply

max0n

10 months ago, # | ☆ 0

Thanks for the problems. Regarding problem C: how does the last paragraph explain that achievable m values form a range? Could someone clarify? I am thinking of different proof: Consider a string of form C...CB...BA...A that has maximum possible value of m (as in editorial A <= B <= C). Then take A or B and put it between some 2 letters C. This decreases number of equal adjacent pairs by 2. If one needs to decrease number of pairs by odd number, just put the A or B at the beginning of the sequence. It will decrease number of adjacent pairs by one.

→ Reply

godsdice

10 months ago, # ^ | ☆ 0

You need to start with the string where m=0. Then, you need to remove the most recurring character in that string and reform that string for m=0. Then, you need to append the removed character and repeat this approach for the remaining string. In your example string CCBBAA, for m=0, we will first form ABCABC. Suppose we remove one C here and reform the string for m=0, we get CABABA. Appending the removed C, we get CCABAB. Now, we need to repeat this approach for the rest of the string: ABCABC, m = 0 CCABAB, m = 1 CCBAAB, m = 2 CCBBAA, m = 3

→ Reply

Hogwarts.dropout

10 months ago, # | ☆ +3

someone please explain graphical approach of D.

→ Reply

Nikhil_knk

10 months ago, # ^ | ☆ 0

Here the nodes are considered as the banned builds... so it is basically a BFS swapping every single element of that banned build to get a new node... I guess this is what they meant to say about the graph approach.

→ Reply

Mansurov_Muhammadamin

10 months ago, # | ☆ 0

Hello I'm newbie can someone write code for first example in c++ thank you

→ Reply

clairvoyant

10 months ago, # | ☆ 0

In B,

```
"Now build the string for m=0"
```

How to prove that we can always build for m = 0?

→ Reply

float_Sammy

10 months ago, # ^ | ☆ +5

This means we have to try to build the string with no repetition, for ex if we are given input as 5A 2B 1C and we try to build the string for m = 0. the string should look like this A__A__A__A__A as you can see we have 4 empty spaces but not enough B's and C's to make a string with no same adjacent letters.

→ Reply

Nikhil_knk

10 months ago, # | ☆ +5

Using banned builds to get good build is crazy idea[problem:D]

→ Reply

10 months ago, # | ☆ ← Rev. 2 +7

I have a different solution for D.

YPK

It's based on the fact that if you have two arrays a and b of lengths n1 and n2, you can build the largest m+1 sum pairs (out of n1*n2 pairs) using binary search. You can do this sequentially for every array or use divide and conquer to find all largest m+1 possible builds. One of them must be the answer as only m builds are banned.

Here is the implementation.

→ Reply

10 months ago, # | ☆ ← Rev. 2 0

i wonder if i can solve problem C in another way:

first find the maximum value in array and the sum of the array.Call them m and s

practise_makes_perfect

then,if $m < x$,then the cost will be $x - m$.

if $s - m < y$then the cost will be $y - (s - m)$.

the final answer will be the sum of the cost.

Why binary search?

→ Reply

zeratul_34

10 months ago, # | ☆ ← Rev. 2 -13

。

→ Reply

practise_makes_perfect

10 months ago, # ^ | ☆ 0

show your code please.

→ Reply

gaurav1903

10 months ago, # | ☆ -18

What does question D even mean?? I didn't understand it

→ Reply

Relief

10 months ago, # | ☆ 0

Problem D. I have implemented the solution in the editorial, however, I'm getting TLE #6. Can anyone please take a look at my solution and figure out the reason for TLE? Thank you.

→ Reply

awoo

10 months ago, # ^ | ☆ +1

You never update your visited set

→ Reply

Relief

10 months ago, # ^ | ☆ 0

OMG! So silly mistake. Thanks!! Giving a long break in competitive programming has shown itself.

→ Reply

micky2100

10 months ago, # | ☆ 0

Please explain the o/p. Like for 1574E — Coloring, I wasn't able to figure how the o/p was reached.

→ Reply

EIragoIEl3enab

10 months ago, # | ☆ +18

How can we solve problem F using FFT?

→ Reply

Rishabh.25

10 months ago, # | ☆ 0

https://codeforces.com/contest/1574/submission/129811930 GETTING TLE for test case 6

→ Reply

bps

10 months ago, # | ☆ 0

I am have understood B and submitted it. But I want to try to print any string of that type if it exists. I m not sure how to implement it. Need little help. I don't want to multiple if-else conditions.

→ Reply

niteshxyz

10 months ago, # | ☆ 0

can anybody help me in slay the dragon problem i am getting WA in test case 3 https://codeforces.com/contest/1574/submission/129995361

→ Reply

changoi

10 months ago, # | ☆ +5

If someone found the editorial solution of E trickier to understand, probably this might help.

Convention:

- Stripe:- Band of alternate 0 & 1, eg: 0101010101.... or 1010101010...
- RowStripe0: A stripe that starts with 0, and forms the row of the grid. Eg: 010101...
- RowStripe1: A stripe that starts with 1, and forms the row of the grid. Eg: 101010...
- Similarly, ColStripe0, ColStripe1
- n: number of rows, m: number of columns
- chess pattern: grid with chessboard pattern(first cell white)
- anti chess pattern: grid with inverted chessboard pattern(first cell black)

Observations:

O1: A valid grid consists of either

- Combination of n RowStripes: 2^n ways
- or Combination of m ColStripes: 2^m ways
- chess & anti chess patterns can be obtained by both 1) & 2)
  - # ways to form valid grids, when all cells are empty = 2^n + 2^m - 2

O2: As we can see we can segregate the answer into two-part,

- Grids which are formed by RowStripes
- Grids which are formed by ColStripes Thus, from now on, we treat them separately.

O3: Partially filled grid.

- Say column c has some cells already filled.
  - **Case 1**: c has cells filled according to the chess(or anti chess) pattern only
    - Then, column c can have only one possible configuration. => # ways to form valid grid with ColStripes = 2^(m-1)
    - If such k columns are already filled => # ways to form valid grid with ColStripes = 2^(m-k)
  - **Case 2**: Column c has cells corresponding to both chess and antichess pattern
    - Then, we can't fill the cell c neither with ColStripe0 nor with ColStripe1 => # ways to form valid grid with ColStripes = 0. **c is thus a bad_column**
- Similar derivation if some row r is filled
- Collission case:
  - If the grid was completely empty then there were two collisions, as

- If the grid was completely empty then there were two collisions, as explained in **O1.3**
- If the grid was partially filled
  - Case 1: If grid filled with either chess (or anti chess) pattern => Subtract 1 from final answer
  - Case 2: If grid is filled with both patterns => No subtraction is required, as chess or anti chess pattern cant be achieved together.

Code

- C1 Mark columns that are already filled
  - For each column maintain whether it is filled in ColStripe0 pattern or ColStripe1 pattern
- C2 Similar to C1, mark rows that are filled
- C3 maintain bad_rows, bad_cols, empty_row, empty_cols

See implementation here: 130091893

(Solution was inspired by tourist's submission: 129358674)

→ Reply

ParthDwivedi

10 months ago, # ^ | ☆ 0

Thank you so much!

→ Reply

rakesh9700

10 months ago, # | ☆ ← Rev. 2 0

Can anybody explain how (c-1)-(a+b) <= m <= (a+b+c-3) is taking care of the case when (c-1)<a+b under the above constraint a<=b<=c?

→ Reply

WTY_ZJOI

10 months ago, # | ☆ 0

For the tutorial of problem E, how to understand "If there are two cells of same color in the same row with even number of cells between them then there is the vertical strip (because there are always two adjacent cells with same color between them)?" Can someone help me with an example? Thanks.

In my opinion, there is obviously a counterexample: 1 0 0 1 0 1 1 0 There are no vertical strips?

→ Reply

Astraea

8 months ago, # | ☆ 0

can anyone explain me in the problem B's tutorial: the strength of defend heroes is equal to $\sum_{j=1}^{n} a[j] - a[i]$ but not the sum of remained heroes not involved in killing the dragon?

→ Reply

cftryharder

8 months ago, # | ☆ 0

On problem C:

Why we should only consider those 2 cases? Any proof on this?

→ Reply

bashNewbie

8 months ago, # ^ | ☆ 0

For the second case maximum a? One way might be to look at the two possible cases for $a < x$, $s - a \geq y$ and $s - a < y$ and the costs for each case, and costs against one another.

→ Reply

Sivasi

7 months ago, # | ☆ 0

//for problem no. 1574A //code in c++

include <bits/stdc++.h>

using namespace std; int main() { int t; cin>>t; while(t--){ int n; cin>>n; int n1=n;

using namespace std; int main() { int t; cin>>t; while(t--){ int n; cin>>n; int i=n;

for(int i=0;i<n;i++){ for(int j=0;j<i;j++){ cout<<"()"; } for(int j=0;j<(n-i);j++){ cout<<"
("; } for(int j=0;j<(n-i);j++){ cout<<")"; } cout<<endl; } } return 0; }

→ Reply