CODEFORCES
Sponsored by TON

| Spartan2804 | Logout
You have +252! Wow!

HOME TOP CATALOG CONTESTS GYM PROBLEMSET GROUPS RATING EDU API CALENDAR HELP

YUJIAHE BLOG TEAMS SUBMISSIONS GROUPS CONTESTS PROBLEMSETTING

YuJiahe's blog

Codeforces Round 1035 (Div. 2) Editorial

By YuJiahe, history, 26 hours ago,

Sorry for misjudging the difficulty of problem D, but we still hope you enjoyed the problems!

2119A - Add or XOR

Author: YuJiahe

Hint 1

$a \oplus 1$ is equivalent to $a-1$ when a is an odd number, otherwise it is equivalent to $a+1$.

Hint 2

If you apply $a \leftarrow a \oplus 1$ when a is odd, then that operation should make $a=b$.

Tutorial

2119A - Add or XOR

Obviously, if $b<a$ and $a \oplus 1=b$, the answer is y. If $b<a$ and $a \oplus 1 \neq b$, there is no solution.

After excluding the above, there is $a \leq b$. Then the bitwise XOR operation can only be used if a is even; otherwise, it will cancel out the last operation.

So if $x \leq y$, then always use the add operation; otherwise, use the bitwise XOR operation if a is even, and use the add operation if it is odd.

Let $d=b-a$. The minimal cost is:

$$ans = \begin{cases} \lceil d/2 \rceil \cdot \min(x,y) + \lfloor d/2 \rfloor \cdot x & a \text{ is even} \\ \lfloor d/2 \rfloor \cdot \min(x,y) + \lceil d/2 \rceil \cdot x & a \text{ is odd} \end{cases}$$

Which is computable in $O(1)$. $O(b-a)$ solutions are also passable.

Implementation

```
#include <algorithm>
#include <cstring>
#include <cstdio>
#include <vector>
#include <queue>
using namespace std;
int t, a, b, x, y;
int main(){scanf("%d", &t);
        while (t --){scanf("%d%d%d%d", &a, &b, &x, &y);
                if (a > b) printf("%d\n", (a ^ 1) == b ? y : -1);
                else {int c0 = b - a, c1 = (b + 1 >> 1) - (a + 1 >> 1);
                printf("%lld\n", y > x ? 1ll * c0 * x : 1ll * (c0 - c1) * x + 1ll * c1 * y);}
        }
}
```

2119B - Line Segments

Author: Lyz09

Hint 1

→ Pay attention

Before contest
Codeforces Round (Div. 1 + Div. 2)
12 days

→ Spartan2804

Rating: 945
Contribution: 0

- Settings
- Blog
- Teams
- Submissions
- Favourites
- Talks
- Contests

Spartan2804

→ Top rated

#	User	Rating
1	jiangly	3756
2	tourist	3723
3	orzdevinwang	3696
4	Kevin114514	3647
5	Radewoosh	3631
6	ecnerwala	3596
7	Benq	3527
8	maroonrk	3518
9	ksun48	3484
10	Nachia	3463

Countries | Cities | Organizations View all →

→ Top contributors

#	User	Contrib.
1	errorgorn	169
2	Qingyu	165
3	Dominater069	159
4	cry	158
5	Um_nik	157
6	adamant	155
7	-is-this-fft-	153
8	djm03178	148
9	soullless	147
10	chromate00	143

View all →

→ Find user

Handle:

Find

→ Recent actions

Transform (p_x, p_y) to (q_x, q_y) into an operation where a_i is the distance between the two points.

Hint 2

When $n+1=3$, there is a solution if and only if the largest a_i is less than or equal to the sum of the remaining a_i.

Tutorial

2119B - Line Segments

First, add an additional vector with magnitude equal to the distance d from (p_x, p_y) to (q_x, q_y), directed back to the starting point. This converts the problem to checking if the $n+1$ vectors can sum to zero.

Now, the problem reduces to determining whether these $n+1$ edges can form a polygon (possibly degenerate). The necessary and sufficient condition is that no single edge exceeds the sum of the others. To see why:

- Necessity: If the longest edge m satisfies $m > \sum_{\text{other edges}}$, it cannot connect with the remaining edges.
- Sufficiency: Otherwise, we can arrange the edges:
  1. Group the edges into three segments: the longest m and two groups with sums s_1, s_2 (sums as close as possible to $\frac{S-m}{2}$ where S is total length).
  2. These three segments form a triangle because:
     - $m \leq s_1 + s_2$ (by initial condition).
     - $s_1 \leq m + s_2$ since $s_1 \leq \frac{S-m}{2} + \frac{\max(a_i)}{2} \leq m + s_2$ when $m \geq \max(a_i)$.
     - Similarly, $s_2 \leq m + s_1$.
  3. The degenerate case ($n+1=2$) is covered when $s_1 = 0$ or $s_2 = 0$.

Therefore, we simply check if the maximum edge length $m = \max(\max_i a_i, d)$ satisfies $m \leq S - m$, where $S = d + \sum_{i=1}^{n} a_i$ and d is the Euclidean distance between the starting point and the terminal point.

Implementation 1

```
#include<iostream>
#include<algorithm>
#include<cmath>
using namespace std;
#define N 100010
#define int long long
int t,n,sx,sy,tx,ty;
double a[N];
signed main()
{
        cin>>t;
        while(t--)
        {
                cin>>n;
                cin>>sx>>sy>>tx>>ty;
                for(int i=1;i<=n;i++)
                 cin>>a[i];
                a[++n]=sqrt((sx-tx)*(sx-tx)+(sy-ty)*(sy-ty));
                sort(a+1,a+n+1);
                double sum=a[n];
                for(int i=1;i<n;i++)
                 sum-=a[i];
                if(sum<=0)
                 puts("Yes");
                else
                 puts("No");
```

mingga → free palestine

walaelaa13 → Help me represent Morocco in EGOI 25

Pajaraja → Olympicode Open Olympiad — Editorials and Results

Proof_by_QED → EPIC Institute of Technology Round Summer 2025 (Codeforces Round 1036, Div. 1 + Div. 2) Editorial

chromate00 → Introducing the newest entertainment in Codeforces: Div2FCheaterGuessr!

LGMS_sfls2029 → APPEAL post

ro_.hi__th._ → Weird issue that I am facing.

YuJiahe → Codeforces Round 1035 (Div. 2) Editorial

Errichto → My Streams in Summer 2025

o_e_a_e_o_e_e_e_a_e → CP-128MB Sheet — A New Era of Competitive Programming

Swap-nil → Codeforces Round #956 (Div. 2) and ByteRace 2024 Editorial

bonavara → Connecting this web to lqdoj.edu.vn

nguyentrongtin09 → I LOVE LE THU HANG

Sanae → An unofficial tutorial on 2124G

iez → This is going to get downvoted alot but im looking for a Programming buddy

Proof_by_QED → EPIC Institute of Technology Round Summer 2025 (Codeforces Round 1036, Div. 1 + Div. 2)

__baozii__ → DAG with exactly k distinct paths from vertex 1 to n

synthborne → sir is so strong

AlperenT → Teams Going to the 2025 ICPC World Finals Baku

reirugan → Codeforces Round 1034 (Div. 3) Editorial

atcoder_official → AtCoder Beginner Contest 412 Announcement

chen_zhe → A very brief introduction to Olympiad in Informatics in mainland China

Don_quixxote → Google Intern OA Problems

JaySharma1048576 → Codeforces Round #785 (Div. 2) Editorial

Arpa → GoForGold Long Challenge — June 2025 Editorial

Detailed →

```
        }
}
```

Implementation 2

```
#include<iostream>
#include<algorithm>
using namespace std;
#define N 100010
#define int long long
int t,n,sx,sy,tx,ty,a[N];
signed main()
{
        cin>>t;
        while(t--)
        {
                cin>>n;
                cin>>sx>>sy>>tx>>ty;
                for(int i=1;i<=n;i++)
                 cin>>a[i];
                sort(a+1,a+n+1);
                int p=(sx-tx)*(sx-tx)+(sy-ty)*(sy-ty);
                if(p>a[n]*a[n])
                {
                        int sum=0;
                        for(int i=1;i<=n;i++)
                        {
                                sum+=a[i];
                                if(sum*sum>=p)
                                 break;
                        }
                        if(sum*sum>=p)
                         puts("Yes");
                        else
                         puts("No");
                }
                else
                {
                        int sum=a[n];
                        for(int i=1;i<=n-1;i++)
                         sum-=a[i];
                        if(sum<=0||sum*sum<=p)
                         puts("Yes");
                        else
                         puts("No");
                }
        }
}
```

2119C - A Good Problem

Author: **lizhous** Preparation: **Lyz09**

Hint 1

For odd n, setting $a_i = l$ for all i suffices.

Hint 2

For even n, there does not exist a valid array a that satisfies $a_1 = a_2 = \cdots = a_{n-1} = l$.

Hint 3

For even n, try to make $a_1 = a_2 = \cdots = a_{n-2} = l$.

Tutorial

2119C - A Good Problem

For odd n, setting $a_i = l$ for all i suffices.

Consider the even $n > 2$ (if $n = 2$ then the only legal array is $[0, 0]$, so no solution exists because $l \geq 1$).

To achieve the lexicographically smallest array, we set $a_i = l$ for $i = 1$ to $n - 1$, then try choosing $a_n \in [l, r]$, which would require satisfying $l \,\&\, a_n = l \oplus a_n$. Now to determine whether there is a legal a_n, then consider each bit separately. Since $l \geq 1$, there must exist a bit in the binary representation of l that is 1, and we observe that for any bit where l has 1, $l \,\&\, x \neq l \oplus x$ holds for $x = 0$ and $x = 1$, so we can't just change a_n.

If no valid a_n exists, we adjust both a_{n-1} and a_n. They must satisfy $a_{n-1} \oplus a_n = l \,\&\, a_{n-1} \,\&\, a_n, l \leq a_{n-1}, a_n \leq r$. Per-bit constraints:

- If l's bit is 1: a_{n-1} and a_n must both be 0.
- If l's bit is 0: a_{n-1} and a_n must be equal (both 0 or both 1).

From the per-bit constraints, a_{n-1} and a_n must be equal at every bit, implying $a_{n-1} = a_n$, so $a_{n-1} \oplus a_n = 0$ and $a_{n-1} \,\&\, a_n = a_n$. From $a_{n-1} \oplus a_n = l \,\&\, a_{n-1} \,\&\, a_n$, we have $l \,\&\, a_n = 0$. Finally, we solve for the minimum a_n by greedily going from high to low bit.

Let b be the highest bit set in l. Since a_n must be $\geq l$ but have bit b unset (due to $l \,\&\, a_n = 0$), a_n requires a 1 at some bit higher than b. And bit b is already different, so the lower bits may be set to 0, so eventually a_n is the smallest power of two greater than l that lies in $[l, r]$.

If no solution exists for a_{n-1}, a_n, then l and r share the same highest bit b. The bitwise AND value is 1, and the bitwise XOR value is 0 on this bit, so it is easy to see that there must be $\&_{i=1}^{n} a_i \neq \oplus_{i=1}^{n} a_i$.

Implementation

```
#include<iostream>
using namespace std;
#define int long long
int t,n,l,r,k;
signed main()
{
        cin>>t;
        while(t--)
        {
                cin>>n>>l>>r>>k;
                if(n%2==1)
                {
                        cout<<l<<endl;
                        continue;
                }
                if(n==2)
                {
                        cout<<-1<<endl;
                        continue;
                }
                int res=1;
                bool fl=0;
                while(res<=r)
                {
                        if(res>l)
                        {
                                fl=1;
                                if(k<=n-2)
                                 cout<<l<<endl;
                                else
                                 cout<<res<<endl;
                                break;
                        }
                        res*=2;
                }
                if(!fl)
                 cout<<-1<<endl;
        }
}
```

2119D - Token Removing

Author: **YuJiahe**

Hint 1

Calculating the weight of a single valid sequence is difficult. Try another way?

Tutorial

2119D - Token Removing

Consider determine which tokens will be removed first. Then we just need to count the number of ways that, for each position p_i with a token to be removed, find an closed segment $[l_i, r_i]$ that contains position p_i, and these r_i are different to each other.

Assuming that the positions of the tokens are $n \ge p_0 > p_1 > \cdots > p_{k-1} \ge 1$, the answer to the above is obviously $\prod_{i=0}^{k-1} p_i(n - p_i + 1 - i)$. This inspires the following DP state: let $f_{i,j}$ denote the answer for the sum of $\prod_{i=0}^{k-1} p_i(n - p_i + 1 - i)$ when $p_{k-1} \ge n - i + 1$ and $k = i - j$. Consider whether p_{k-1} is equal to $n - i + 1$, we have:

$$f_{i,j} = f_{i-1,j-1} + (n - i + 1)(j + 1)f_{i-1,j}$$

The answer is $\sum_{j=0}^{n} f_{n,j}$. The total time complexity is $O(n^2)$.

Implementation

```
#include <algorithm>
#include <cstring>
#include <cstdio>
#include <vector>
#include <queue>
using namespace std;
const int N = 5005;
int t, n, mod, f[N][N], ans;
inline void add(int& x, int y){x += y;if (x >= mod) x -= mod;}
int main(){scanf("%d", &t);
        while (t --){scanf("%d%d", &n, &mod), f[0][0] = 1, ans = 0;
                for (int i = 1;i <= n;i ++) for (int j = 0;j <= i;j ++) f[i][j]
= 0;
                for (int i = 1;i <= n;i ++)
                for (int j = 0, now;j < i;j ++)
                if (now = f[i - 1][j]) add(f[i][j + 1], now),
                f[i][j] = (f[i][j] + (n - i + 1ll) * (j + 1) * now) % mod;
                for (int j = 0;j <= n;j ++) add(ans, f[n][j]);printf("%d\n",
ans);
        }
}
```

2119E - And Constraint

Author: **ma2021tyoi0037**

Hint 1

As same as the title, can you find a simple necessary constraint?

Hint 2

Consider $a_i = 0$ first.

Tutorial

2119E - Ограничение на И

First, obviously b_i must include $a_{i-1} | a_i$, which is the fundamental requirement.

Secondly, we can note that there are only 31 possible effective values for b_i:

1. Considering the upper bound of the problem, it is not difficult to increase all b_i to above 2^{29}, then change some bits to above 2^{30}, and then satisfy the previous constraints.
2. Consider how to make the effective values so few.

We might consider which b_i are more likely to be the answer. Apart from the binary bit constraints that must be satisfied, we should try to avoid having 1s in each binary bit as much as possible, because this might cause our sequence not to satisfy the conditions.

Observing the process of increasing b_i, the useful b_i exhibit a stepped pattern: the preceding bits are the same as b_i, the next bit is set to 1, and all subsequent bits are 0. Numbers not on this step are certainly useless.

However, we did not consider the bits that are forced to be included earlier. We can directly apply the bitwise OR with this constraint, and then we find the useful values.

With the useful values, we can directly design $f_{i,j}$ to represent the minimum number of operations when the first i numbers have been determined, and the i-th number becomes j.

During the transition, we can enumerate the value k for the $(i+1)$-th position, and if $j \,\&\, k$ satisfies the constraint, then we can make the transition.

The time complexity of this algorithm is $O(n \log^2 V)$, where V is the maximum value among a_i and b_i.

Implementation

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

typedef long long ll;
struct stu{
        int x;
        ll dp;
};
int n,a[100007],b[100007];
vector<stu> f,g;
const ll inf=0x3f3f3f3f3f3f3f3f;
void Subt(){
        cin>>n;
        for(int i=1;i<n;i++) cin>>a[i];
        for(int i=1;i<=n;i++) cin>>b[i];
        a[n]=0;
        f.clear();
        f.push_back((stu){0,0});
        for(int i=1;i<=n;i++){
                int x=0;
                g.clear();
                for(int j=30;j>=-1;j--){
                        int y=x|a[i-1]|a[i];
                        if(j!=-1) y|=(1<<j);
                        if(y>=b[i]){
                                ll mn=inf;
                                for(stu l:f){
                                        if((l.x&y)!=a[i-1]) continue;
                                        mn=min(mn,l.dp+y-b[i]);
                                }
                                if(mn<inf) g.push_back((stu){y,mn});
                        }
                        x|=((1<<j)&b[i]);
                }
                swap(f,g);
        }
        ll mn=inf;
        for(stu l:f) mn=min(mn,l.dp);
        if(mn<inf) cout<<mn<<endl;
        else cout<<-1<<endl;
        return;
}
```

```
signed main(){
	ios::sync_with_stdio(0),cin.tie(0),cout.tie(0);
	int T=1;
	cin>>T;
	while(T--) Subt();
}
```

2119F - Volcanic Eruptions

Author: **lizhous**

Hint 1

Try to determine the vertex where you were before the last move.

Hint 2

I just need to calculate the earliest time I can reach the end vertex.

Hint 3

What form of moving path from the starting vertex to the end vertex is considered optimal?

Tutorial

2119F - Volcanic Eruptions

We define a "TURN" as consecutively traversing the same edge. Define the endpoint as the vertex reached before the last move. Use (a, b) to define an edge (u, v) satisfied $w_u = a, w_v = b$.

First, if the path does not traverse through the edge $(1, 1)$, the sequence of vertex weights along the path must alternate between 1 and -1. The maximum moves can be computed by enumerating endpoints ed where the path from st to ed satisfies this alternation. Then we can repeatedly increase the number of TURNs at any $(1, -1)$ to ensure you always reach the endpoint just before the lava does. A simple parity check then suffices to compute the answer for this endpoint.

The remaining paths must include $(1, 1)$. We observe that performing TURNs on more than one distinct edge is suboptimal. This is because all TURNs can be adjusted to occur on the first traversal of the $(1, 1)$ edge. Specifically, if the current path involves x TURNs, we can modify it to perform all x TURNs on the first traversal of $(1, 1)$ and then remove all remaining TURNs. This adjustment does not affect the validity of the path:

1. Path continuity: The start and end vertices of a TURN are the same, so adding or removing a TURN does not break the path's continuity.

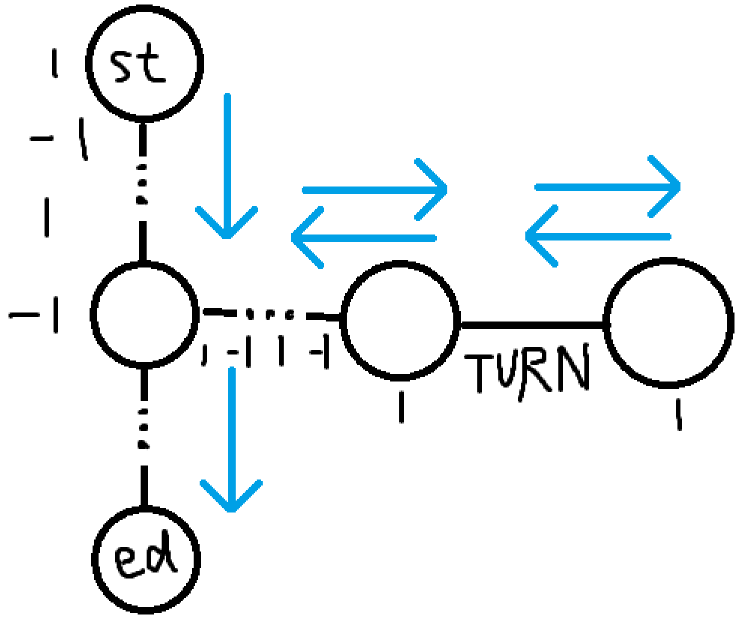
2. Death by lava: Since your speed matches the lava's, as long as you are not in the lava at the end, you won't encounter it midway.

3. Death by low health: Before the first $(1, 1)$, the path must alternate between 1 and -1. Removing TURNs in this segment won't cause death. Moreover, since the number of TURNs at this $(1, 1)$ is no fewer than in the original plan, the health at the vertices after TURNs will be no lower than in the original plan, ensuring no death.

If the endpoint is ed, then based on the conclusion that TURNs occur only on one edge, the optimal path must consist of a main path from the start st to ed with a TURN branch attached. This branch must satisfy:

— Its outermost edge is $(1, 1)$.

— The path leading to this edge alternates between 1 and -1.



For a given ed, to reach the endpoint earlier, we need to find the nearest $(1, 1)$ to the main path where the segment from the start to it alternates between 1 and -1. We can perform a BFS starting from each $(1, 1)$, selecting the next vertex under the condition that its weight w differs from the current vertex's. This allows us to preprocess the nearest $(1, 1)$ for each vertex, which can then be directly referenced when constructing the main path.

For a given ed, the vertex with the minimum health is also determined, since the TURN branch (alternating between 1 and -1) does not affect the minimum health vertex. Thus, ensuring health remains above zero at all times reduces to ensuring the minimum health vertex satisfies this condition.

For a given ed, all constraints can be easily resolved to compute the earliest arrival time at the endpoint. Since your speed matches the lava's, we only need to compare your arrival time with the lava's to determine if ed is a valid endpoint.

After fixing the endpoint, we can repeatedly increase the number of TURNs at $(1, 1)$ to ensure you always reach the endpoint just before the lava does. A simple parity check then suffices to compute the answer for this endpoint.

The final answer is the maximum value obtained across all possible endpoints.

Implementation

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<vector>
#include<set>
#include<queue>
#include<unordered_map>
#include<map>
#define int long long
using namespace std;
int n,st,dist[1000002],w[1000001],tim[1000001],ans;
vector <int> g[1000001];
int f[1000001];
queue<int> q;
void getdis(int u,int fa)
{
        for(int v:g[u])
        {
                if(v==fa) continue;
                dist[v]=dist[u]+1;
                getdis(v,u);
        }
}
```

```
void dfs(int u,int fa,int k,bool inn,int hei,int TIM,int dis)
{
        if(hei<0)
        {
                TIM=max(TIM,(-hei+1)/2*2+k*2+dis);
        }
        if(TIM<=dist[u])
        {
                ans=max(ans,dist[u]);
        }
        else
        {
                return;
        }
        for(int v:g[u])
        {
                if(v==fa) continue;
                if(inn&&w[u]!=w[v])
                {
                        dfs(v,u,min(k,tim[v]),inn,hei+w[v],TIM+1,dis+1);
                }
                else
                {
                        dfs(v,u,k,0,hei+w[v],TIM+1,dis+1);
                }
        }
}
int T;
signed main()
{
        ios::sync_with_stdio(false);
        cin>>T;
        int cnt=0;
        while(T--)
        {
                cnt++;
                cin>>n>>st;
                for(int i=1;i<=n;i++)
                {
                        dist[i]=0;
                        tim[i]=1000000000;
                        g[i].clear();
                        f[i]=-100000000000000000;
                }
                ans=0;
                for(int i=1;i<=n;i++)
                {
                        cin>>w[i];
                }
                for(int i=1,u,v;i<n;i++)
                {
                        cin>>u>>v;
                        g[u].push_back(v);
                        g[v].push_back(u);
                }
                getdis(1,-1);
                for(int i=1;i<=n;i++)
                {
                        for(int v:g[i])
                        {
                                if(w[i]==1&&w[v]==1)
                                {
                                        tim[i]=0;
                                        q.push(i);
                                        break;
                                }
```

```
            }
        }
        while(!q.empty())
        {
            int u=q.front();
            q.pop();
            for(int v:g[u])
            {
                if(tim[v]==1000000000&&w[v]!=w[u])
                {
                    tim[v]=tim[u]+1;
                    q.push(v);
                }
            }
        }
        dfs(st,-1,tim[st],1,w[st],1,1);
        cout<<ans+(dist[st]&1)-1<<'\n';
    }
}
```

Tutorial of Codeforces Round 1035 (Div. 2)

+52 ☆ YuJiahe 26 hours ago 30

Comments (30)

Write comment?

YuJiahe

22 hours ago, hide # | ☆ 0

*Auto comment: topic has been updated by **YuJiahe** (previous revision, new revision, compare).*

→ Reply

ma2021tyoi0037

22 hours ago, hide # | ☆ -22

D is interesting and easy to code.

→ Reply

Badri41

22 hours ago, hide # | ☆ 0

ABC were easy

D was just chinese for me. Hope to learn and be better for next contest. Learnt about the reverse the Summation Technique.

Thanks for the problems :)

→ Reply

zac2010

20 hours ago, hide # ^ | ☆ 0

An interesting thing is, I thought you were referring to the AtCoder Beginner Contest.

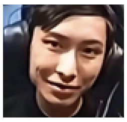
→ Reply

zeyd123

22 hours ago, hide # | ☆ +8

I see that problems so hard, and editorial too

→ Reply

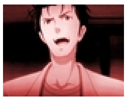


Wantingfire

3 hours ago, hide # ^ | ☆ 0

Me too and i only finished A during the contest……

→ Reply



22 hours ago, hide # | ☆ 0

Can someone explain the solution to E in simpler words? "preceding bits", "next bit"