HUT三面题解

Problem A

明白三个运算符后可以发现我们想要c最大一定是$a\ or\ b$,这样只要任何二进制位上a和b有一个为1，c在这个二进制位上就一定为1，由于$1e9$不超过2^{30},所以我们从第30位开始遍历，依次找到l,r，最后算出距离

```
#include<bits/stdc++.h>
using namespace std;
void solve()
{
        int a,b;cin>>a>>b;
        int c=a|b;
        int l=0,r=0;//r表示最大位，l表示最小位
        //从最大位开始遍历
        for(int i=30;i>=0;i--){
                if(c&(1<<i)){//判断c这一位是否为1
                        if(r==0) r=i;//最大的二进制位为r
                        l=i;//l是找最小位，所以每次直接赋值即可
                }
        }
        printf("%d\n",r-l+1);
}
int main()
{
        ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int tt;tt=1;
        //cin>>tt;
    while(tt--){
        solve();
    }
    return 0;
}
```

Problem B

本题中小津只能选择行动，因为如果小津选择不移动，则我可以向小津靠近，这对小津显然是不利的。初看题面可能最直接的思路就是我怎么向小津靠近，而小津需要如何远离我并前往樋口师傅家，但这会将问题变得很复杂，所以可以换个思路找更简单的方法。容易发现小津在前往（n,m）的前一个点只有（$n-1,m$）和（$n,m-1$）两处，所以我只需要待在（$n-1,m-1$）处即可，只要小津到达（$n-1,m$）或（$n,m-1$），我都可以直接抓住小津。

接下来理性地验证一下这个方法的可行性。小津到达（$n-1,m$）或（$n,m-1$）前一个点需要花费的行动次数是$T_{oz}=n+m-2$，我到达（$n-1,m-1$）需要花费的行动次数是$T_{我}=n+m-2-$（$x+y$）。由于小津先进行移动，所以只有当$T_{oz}\geq T_{我}$时才能到达樋口师傅家，也就是说需要满足x+y==0,即我的初始坐标为（0,0）。这时我可以直接抓住小津。所以小津无法到达樋口师傅家。

感性地证明一下，显然，我的路程比小津的短，所以我一定能比小津提前到。

综上，这题直接输出“Go to hell,oz!”即可。

Problem C

显然题目中多个重复的元素可以视为一个元素，将数组a中的元素去重后得到一个数组f

设 f_i 表示第 i 号点被选中的次数（显然只可能是 0 或 1）。

则要求的答案就是 $E[\sum f_i]$，即 f_i 的和的期望，

由期望的可加性，有 $E[\sum f_i]=\sum E[f_i]$。

由于 f_i 的取值只能是 0 或 1，则 f_i 的期望就等于 f_i 等于 1 的概率，设为 p_i。

则答案为 $\sum p_i$，即每个节点被选中的概率的和。 考虑随机生成一个操作序列，找到序列中第一个未被染色的节点，并染色这个节点和小于它所有节点，重复这个操作，直到序列中所有节点都被染色。

那么 i 号节点被选中的前提就是在序列中，小于i节点的节点都在 i 号节点的后面。否则，就会在选择 i 号节点之前选择小于i的节点，并且 i 节点就会被染色，而不会被选中。

设i为第x_i小的节点,那么大于等于i号节点的节点共有 x_i个，故 i 号节点在这个随机序列中排在所有祖先前面的概率是 $1/x[i]$，故 i 号节点被选中的概率是 $1/x[i]$。

故答案为 $\sum 1/x[i]$。

即为$1+1/2+1/3+...+1/m$，m为数组中不同元素个数

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

double qwq(int n, vector<int>& arr) {
    std::vector<int> tmp(arr);
    std::sort(tmp.begin(), tmp.end());
    tmp.erase(std::unique(tmp.begin(), tmp.end()), tmp.end());
    for (int i = 0; i < n; ++i)
        arr[i] = std::lower_bound(tmp.begin(), tmp.end(), arr[i]) - tmp.begin();
    double ans = 0;
    for (int i = 1; i <= tmp.size(); i++) ans += 1.0 / i;
    return ans;
}

int main() {
    int n;
    cin >> n;
    vector<int> a(n);
    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }
    double ans = qwq(n, a);
    printf("%.10lf",ans);
    return 0;
}
```

Problem D

最开始我们要把所有容量加起来特判是否小于x，如果小于就按题目要求输出-1 -1。 求用最少的背包数一定是先从容量大的背包开始使用，所以我们只需要先排序然后从后往前遍历即可。求用最多的背包我们只需要注意娃娃数和背包数的大小，如果娃娃数小于背包数，答案就是娃娃数，否则答案就是n个背包。

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e3+10;
int a[N];
void solve()
{
        int n,x;cin>>n>>x;int sum=0;
        for(int i=1;i<=n;i++) cin>>a[i],sum+=a[i];
        //装不下直接输出-1
        if(x>sum){
                printf("-1 -1\n");
                return;
        }
        sort(a+1,a+1+n);//排序
        int mi,ma;
        ma=min(x,n);//背包个数不能超过娃娃个数，所以要取min
        //从后容量最大的背包开始往前遍历
        for(int i=n;i>=1;i--){
                x-=a[i];
                if(x<=0){
                        mi=n-i+1;
                        break;
                }
        }
        printf("%d %d\n",mi,ma);
}
int main()
{
        ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int tt;tt=1;
        //cin>>tt;
    while(tt--){
        solve();
    }
    return 0;
}
```

Problem E

容易发现，本题需要从a_1到a_{n-2}选择$k-1$个数使它们的和最小，同时这$k-1$个数不能相邻。因此可以考虑使用动态规划来解决这个问题。
设$dp[i][j]$为前i个数中选择j个不相邻的数的和的最小值。每个数字有选择和不选两种情况，由此可以写出状态转移方程$dp[i][j]=min(dp[i-2][j-1]+a[i],dp[i-1][j])$。

```
#include<iostream>
#include<cstring>
using namespace  std;

int  main()
{
    int n,k;
    cin>>n>>k;
    int  a[n+1];
    for  (int  i=1;i<=n;++i)
        cin>>a[i];

    int  dp[n+1][k+1];
    memset(dp,0x3f,sizeof  dp);
    for  (int  i=0;i<=n;++i)
        dp[i][0]=0;

    for  (int  i=2;i<=n-1;++i)
    {
        for  (int  j=1;j<k;++j)
        {
            dp[i][j]=min(dp[i-1][j],dp[i-2][j-1]+a[i]);
        }
    }

    cout<<dp[n-1][k-1]<<endl;

    return  0;
}
```

Problem F

$n!$中后面的0的数量为$n!$中因子10的个数，而$10=2*5$，因此转换成求$n!$中质因子2和5的个数的最小值。
所以$n!$的质因子中5的个数是大于等于2的，因此答案为$n!$的质因子中5的数量。

```
#include<iostream>
using namespace std;

int main()
{
    int n;
    cin>>n;

    int ans=0;
    for (int i=5;i<=n;++i)
    {
        for (int j=i;j%5==0;j/=5)
        {
            ++ans;
        }
    }

    cout<<ans<<endl;

    return 0;
}
```

Problem G

签到题，比较双方三个值之和的大小即可解决。由于数据范围较大，注意开longlong。

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

int main(){
    long long s1=0,s2=0;

    for(int i=1;i<=3;i++){
        long long x;cin>>x;
        s1+=x;
    }

    for(int i=1;i<=3;i++){
        long long x;cin>>x;
        s2+=x;
    }

    if(s1>s2) cout<<"Manbo"<<endl;
    if(s1<s2) cout<<"Muri"<<endl;

    return 0;
}
```

Problem H

注意攻击顺序从左到右，当前敌人生命值$h_i \leq 0$才能攻击下一个人。
不难发现，T每增加3可以让h_i减少5
那么这三组行动就会重复$\lfloor h_i/5 \rfloor$组，按照这规律即可解决此题。

```cpp
//静水流深 感恩 I can do all things
#include <bits/stdc++.h>
using namespace std;
#define endl '\n'
#define lowbit(x) x&(-x)
#define int long long
typedef tuple<int,int,int> tup;
typedef pair<int,int> PII;
typedef long long ll;
const double pai=acos(-1.0);
int dx[]={-1,0,1,0};
int dy[]={0,-1,0,1};
const int N=2e5+10;
int a[N];
void solve()
{
        int n;cin>>n;
        ll t=0;
        for(int i=1;i<=n;i++) cin>>a[i];
        for(int i=1;i<=n;i++){
                int mod=t%3;
                if(mod==2){
                        a[i]-=3;
                        t++;
                }
                else if(mod==1){
                        t++,a[i]--;
                        if(a[i]<=0) continue;
                        a[i]-=3;
                        t++;
                }
                if(a[i]<=0) continue;
                mod=a[i]%5;
                if(mod==0) t+=a[i]/5*3;
                else if(mod==1||mod==2) t+=a[i]/5*3+mod;
                else t+=a[i]/5*3+3;
        }
        cout<<t<<endl;
}
```

```
signed main()
{
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int n;
    n=1;
    while(n--){
        solve();
    }
    return 0;
}
```

Problem I

一道很直白的模拟题，我们考虑以下三种情况：

1. 白天为'Y'并且晚上为'Y'，这时把幸福度+3就好，注意在白天有梦时不能把晚上的梦移到白天做；
2. 白天为'Y'而晚上为'N',此时幸福度+2，也很明显；
3. 白天为'N'而晚上为'Y'，这时我们把晚上的梦移到白天做，所以幸福度+2.

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

void solve() {
	int n;
	cin >> n;
	string s1;
	cin >> s1;
	string s2;
	cin >> s2;
	int ans = 0;
	for (int i = 0; i < n; i++) {
		if (s1[i] == 'Y')
			ans += 2;
		if (s2[i] == 'Y') {
			if (s1[i] == 'N')
				ans += 2;
			else
				ans += 1;
		}
	}
	cout << ans << '\n';
}


int main() {
	int _ = 1;
	//cin >> _;
	while (_--)
		solve();

	return 0;
}
```

Problem J

> 题目大意:
> 给定n个物品，你拥有m元，对于第i个物品，价值为a_i

> 构造一种购买方案，使得你购买的物品的价值的极差不能超过1且总价格不能超过m，求你能购买的最大价值。

考虑枚举，若花店中出现有$x+1$花瓣数的花，就从1开始枚举购买一定数量花瓣数为x的花，用剩下的钱去购买花瓣数为$x+1$的花，同时更新最大值。

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

using u32 = unsigned;
using i64 = long long;
using u64 = unsigned long long;

void solve() {
    int n;
    i64 m;
    std::cin >> n >> m;

    std::map<int, int> f;
    for (int i = 0; i < n; i++) {
        int a;
        std::cin >> a;
        f[a]++;
    }

    i64 ans = 0;
    for (auto [x, y] : f) {
        ans = std::max(ans, 1LL * x * std::min<i64>(y, m / x));
        if (f.contains(x + 1)) {
            int z = f[x + 1];
            for (int i = 1; i <= y && 1LL * i * x <= m; i++) {
ans=std::max(ans, 1LL * i * x + 1LL * (x + 1) * std::min<i64>(z, (m - 1LL * i * x) / (x + 1)));
            }
        }
    }
    std::cout << ans << "\n";
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);

    int T;
    std::cin >> T;

    while (T--) {
```

```
        solve();
    }

    return 0;
}
```