$Good\quad Key, Bad\quad Key$

- chenjiuri_goodkeybadkey

```
现在有n把钥匙，要打开n个箱子。
分别有两类，good and bad key;
使用good 有一定的花费。但是对箱子中的金币数目没有影响。
使用bad  没有花费，后面的所有宝箱中的金币减半。
问最大钥匙分配得到最大收益。
```

$20mins$

```
问题抽象为一个，分配问题，对于一个宝箱：
对于一个箱子就是两个方案。最终的方案个数非常的多。
考虑一个小规模的问题。
f[i][j]前i个的使用几次旧钥匙的最大收获。
f[i][j]=max(f[i-1][j]+a[i]/j,f[i-1][j-1]+a[j]/j);
所以f[i][j]=max(f[i-1][j],f[i-1][j-1])=max(f[i-2][j-1],f[i-2][j-2],f[i-2][j-2],f[i-2][j-3])
这样转移到的复杂度依然高达$O(n)$
```

$slove$

- 随便构造任意一个解，发现，它可以迁移到一个更优的$bad\quad key$使用数量相同的更优解。
  - 简单证明如下：
    - 形如 g g g g b g;总贡献。$sum_{1\ldots4}+a_5/2+a_6/2-k$
    - 迁移 g g g g g b：总贡献：$sum_{1\ldots4}+a_5+a_6/2-k;$
    - 可见g放在前面最优。
  - 但是注意，这里并不满足二分的条件。
- 实现上：
  - 由于$log_2 1e9\approx 30$因此直接暴力枚举即可。
  - 复杂应该为 $n\times log_2 a$

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn = 2e5 + 10;

ll a[maxn];

void solve()
{
    ll n, k;
    cin >> n >> k;
    ll sum = 0;
    ll ans = 0;
    for (int i = 1; i <= n; i++)
        cin >> a[i];
    a[0] = k;
```

```
    for (ll i = 0; i <= n; i++) //从这一个开开始后面全部选择
    {
        sum += a[i] - k;
        ll now = sum;
        for (int j = i + 1; j <= min(n, i + 32); j++)
            now += (a[j] >> (j - i));
        ans = max(now, ans);
    }
    cout << ans << '\n';
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(nullptr), cout.tie(nullptr);
    int t;
    cin >> t;
    while (t--)
        solve();
}
```

生长思考

- 构造解，然后经典的迁移，看是否可以寻求出一种规律的解集。从而大幅度的优化解空间。