Tricks

1. Memory optimization of bitset solutions.
2. Square root optimization of knapsack/"$3k$ trick": Assume you have n rocks with nonnegative integer weights $a_1, a_2, \ldots, a_n$ such that $a_1 + a_2 + \ldots + a_n = m$.You want to find out if there is a way to choose some rocks such that their total weight is w.Suppose there are three rocks with equal weights a,a,a.Notice that it doesn't make any difference if we replace these three rocks with two rocks with weights $a, 2a$.We can repeat this process of replacing until there are at most two rocks of each weight. The sum of weights is still m,so there can be only $\sqrt{m}$ rocks(see next point).
3. Number of unique elements in a partition: Assume there are n nonnegative integers $a_1 + a_2 + \ldots + a_n = m$,Then there are only $O(\sqrt{m})$ distinct values.
4. Removing elements from a knapsack:

```
Adding a new item is classical:

# we go from large to small so that the already updated dp values won't affect any calculations
for (int i = dp.size() - 1; i >= weight; i--) {
    dp[i] += dp[i - weight];
}
```

```
To undo what we just did, we can simply do everything backwards:

# this moves the array back to the state as it was before the item was added
for (int i = weight; i < dp.size(); i++) {
    dp[i] -= dp[i - weight];
}
```

5. $O(n^2)$ complexity of certain tree DPs:

```
function calc_dp (u):
    for each child v of u:
        calc_dp(v)
    for each child v of u:
        for each child w of u other than v:
            for i in 0..length(dp[v])-1:
                for j in in 0..length(dp[w])-1:
                    # calculate something
                    # typically based on dp[v][i] and dp[w][j]
                    # commonly assign to dp[u][i + j]

calc_dp(root)
```

6. $O(n \times k)$ complexity of certain tree DPs: Suppose instead of the vector being the length of the subtree of u,it is the minimum of k and the length of the subtree of u.

```
function calc_dp (u):
    for each child v of u:
        calc_dp(v)
```

```
    dp[u]=[0]
    for each child v of u:
        temp=[0,0,...,0]
        for i in 0..length(dp[u])-1:
            for j in in 0..length(dp[v])-1:
                if i+j<K:
                    # calculate something
                    # typically based on dp[u][i] and dp[v][j]
                    # commonly assign to temp[i + j]
        pop elements from temp until length(temp)<=K
        dp[u]=temp

calc_dp(root)
```

7. $O(n)$ complexity for some tree DPs: If you merge two subtrees in $O(min(depth_1, depth_2))$,you get exactly $n - treeDepth$ operations in total.This is because every node is merged exactly once! We can imagine that when states in a are merged into b, they just disappear.
8. How to precompute inverse:

```
for (int i = 1; i < N; ++i) {
    inv[i] = (i == 1) ? 1 : mod - 1ll * inv[mod % i] * (mod / i) % mod;
}
```

9. Formula and tips:
    1. there are $O(\frac{n}{log(n)})$ primes up to n.
    2. the nth primes is $O(n \times log(n))$.
    3. $a \equiv b \pmod p \iff p \mid b - a$
    4. $a \equiv b \pmod m, a \equiv b \pmod n \rightarrow a \equiv b \pmod{[m, n]}$
    5. $(k, m) = d, ka_1 \equiv ka_2 \pmod m \rightarrow a_1 \equiv a_2 \pmod{\frac{m}{d}}$
    6. $k \times C_n^k = n \times C_{n-1}^{k-1}$
    7. $C_k^n \times C_m^k = C_m^n \times C_{m-n}^{m-k} \ (m - k < m - n)$
    8. $\sum_0^n C_n^i = 2^n$
    9. $\sum_{k=0}^n (-1)^k \times C_n^k = 0$
    10. $C_n^k + C_n^{k+1} = C_{n+1}^{k+1}$
    11. $\sum_{k=0}^m C_{n+k}^k = C_{n+m+1}^m$

12. $Tolerance:$

由三圆图可得公式：$S_1 \cup S_2 \cup S_3 = S_1 + S_2 + S_3 - (S_1 \cap S_2 + S_1 \cap S_3 + S_2 \cap S_3) + S_1 \cap S_2 \cap S_3$。

推广可得：

$$\bigcup_{i=1}^{m} S_i = \sum_{i=1}^{m} S_i - (S_1 \cap S_2 + S_1 \cap S_3 + \cdots + S_{m-1} \cap S_m) + \cdots + (-1)^{m-1} \bigcap_{i=1}^{m} S_i$$

对于集合 S 的每个元素 x，有 $x, k(1 \leq k \leq n)$，

$cnt_x = C_k^1 - C_k^2 + C_k^3 + \cdots + (-1)^{k-1} C_k^k = 1$。

可得容斥原理的正确性，而根据组合数恒等式推广可知：

$$\sum_{k=0}^{n} (-1)^k C_n^k = 0$$

将等式 cnt_x 两边同减去 $C_k^0 = 1$ 即可得到上式。

10. Series:
    1. $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$
    2. $ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots$
    3. $\sqrt{1+x} = 1 + \frac{x}{2} - \frac{x^2}{8} + \frac{2x^3}{32} - \frac{5x^4}{128} + \ldots$
    4. $\frac{1}{1-x} = \sum_{i \geq 0} x^i$
    5. $\frac{1}{1-ax} = \sum_{i \geq 0} a^i x^i$
    6. $\frac{1}{(1-x)^k} = \sum_{i \geq 0} \binom{i+k-1}{i} x^i$
    7. $(1+x)^k = \sum_{n \geq 0} \binom{k}{n} x^n$
    8. $log(P(x)) = \int \frac{P'(x)}{P(x)}$
11. 欧拉序求LCA:

    $LCA(u,v) = RMQ(first(u), first(v))$
12. MoTree:

    dfs 一棵树，然后如果 dfs 到 x 点，就 `push_back(x)`，dfs 完 x 点，就直接 `push_back(-x)`

    新加入的值是 x ---> `add(x)`

    新加入的值是 - x ---> `del(x)`

    新删除的值是 x ---> `del(x)`

    新删除的值是 - x ---> `add(x)`

    对于u和v，假设$in(u) < in(v)$:

    若$LCA(u,v) == u$，则为$in(u)$到$in(v)$这段区间。

    若$LCA(u,v)! = u$，则为$out(u)$到$in(v)$，需要额外加上$LCA(u,v)$的贡献。

    ❗ ❗ (括号序 $\neq$ 欧拉序) ❗ ❗
13. LCS：将S_1中的字符替换为在S_2中所有出现的下标(按照降序)，转化为LIS。
14. LIS：树状数组/二分优化到$O(nlogn)$。
15. 第一类斯特林数(无符号)

    长度为n的排列构成m个圆(非空轮换)的方案数，记作$S_u(n,m)$。

    递推式：$S_u(n,m) = S_u(n-1,m-1) + S_u(n-1,m) * (n-1)$。

    边界：$S_u(n,0) = [n == 0]$。
16. 第二类斯特林数

把n个不同的数划分为m个集合的方案数，要求不能为空集，记作$S(n,m)$。

递推式：$S(n,m)=S(n-1,m-1)+S(n-1,m)\times m$。

17. 整数划分

一个正整数n写成多个大于等于1且小于等于其本身的整数的和,其中各加数构成的集合为n的一个划分。

递推式：$f(n,m)=f(n,m-1)+f(n-m,m)$

18. 卡特兰数

进栈出栈顺序，对角线，三角形划分，n个节点构成不同的二叉树……

递推式：$H_{n+1}=\sum_{i=0}^{n}H_iH_{n-i}$

通项：$H_n=\frac{C_{2n}^{n}}{n+1}$

19. 拉格朗日插值

$f(x)=\sum_{i=1}^{n}y_i\prod_{j\neq i}\frac{x-x_j}{x_i-x_j}$

20. 四边形不等式

区间包含单调性：若$l\leq l'\leq r'\leq r$，则$w(l',r')\leq w(l,r)$。

四边形不等式：交叉不大于包含，$w(l,r')+w(l',r)\leq w(l,r)+w(l',r')$。

由四边形不等式可推导出$1D/2D$决策单调性。