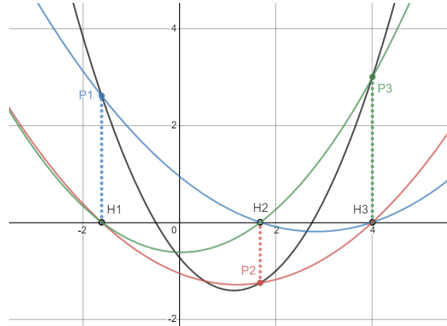


拉格朗日插值

Lagrange Interpolation

Idea: 给定 n 个点 $P_i(x_i, y_i)$, 求过这 n 个点的 $n-1$ 次多项式 $f(x)$.



我们对于每个 i , 找到一个多项式函数 $g_i(x)$ 使之过 P_i 和所有 $j \neq i$ 的 $(x_j, 0)$. 容易知道, $g_i(x)$ 可以如下构造:

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

这样就可以有 $g_i(x_j) = 0$ ($j \neq i$) 以及 $g_i(x_i) = y_i$.

现在把所有 $g_i(x)$ 加起来, 就得到了 $f(x)$:

$$f(x) = \sum_{i=0}^n g_i(x) = \sum_{i=0}^n y_i \prod_{\substack{j=1 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j}$$

因为, 如此有: $f(x_j) = \sum_{i=0}^n g_i(x_j) = y_j$, 也即 $f(x)$ 过所有点 $P_i(x_i, y_i)$.

Implementation: 分子和分母分别计算后再算逆元相乘, 复杂度的瓶颈就不会在求逆元上。

Complexity: $O(n^2)$

Code:

```
1  int n, k;
2  pair<int, int> p[N];
3
4  int main(){
5      scanf("%d%d", &n, &k);
6      for(int i = 1; i <= n; i++){
7          scanf("%d%d", &p[i].first, &p[i].second);
8      }
9      int ans = 0; // ans = f(k)
10     for(int i = 1; i <= n; i++){
11         int up = p[i].second, dn = 1;
12         for(int j = 1; j <= n; j++){
13             if(j == i) continue;
14             up = 1ll * up * (k - p[j].first) % MOD;
15             dn = 1ll * dn * (p[i].first - p[j].first) % MOD;
16         }
17         if(up < 0) up += MOD;
18         if(dn < 0) dn += MOD;
19         ans = (ans + 1ll * up * fpow(dn, MOD-2) % MOD) % MOD;
20     }
21     printf("%d\n", ans);
22     return 0;
23 }
```