Distributing DP

**D - Leaping Tak**[Editorial](https://atcoder.jp/contests/abc179/tasks/abc179_d/editorial)https://img.atcoder.jp/assets/top/img/flag-lang/ja.png / https://img.atcoder.jp/assets/top/img/flag-lang/en.png

Time Limit: 2 sec / Memory Limit: 1024 MB

Score : 400400 points

**Problem Statement**

There are NN cells arranged in a row, numbered 1,2,…,N1,2,…,N from left to right.

Tak lives in these cells and is currently on Cell 11. He is trying to reach Cell NN by using the procedure described below.

You are given an integer KK that is less than or equal to 1010, and KK non-intersecting segments [L1,R1],[L2,R2],…,[LK,RK][L1,R1],[L2,R2],…,[LK,RK]. Let SS be the union of these KK segments. Here, the segment [l,r][l,r] denotes the set consisting of all integers ii that satisfy l≤i≤rl≤i≤r.

* When you are on Cell ii, pick an integer dd from SS and move to Cell i+di+d. You cannot move out of the cells.

To help Tak, find the number of ways to go to Cell NN, modulo 998244353998244353.

**Constraints**

* 2≤N≤2×1052≤N≤2×105
* 1≤K≤min(N,10)1≤K≤min(N,10)
* 1≤Li≤Ri≤N1≤Li≤Ri≤N
* [Li,Ri][Li,Ri] and [Lj,Rj][Lj,Rj] do not intersect (i≠ji≠j)
* All values in input are integers.

**Input**

Input is given from Standard Input in the following format:

NN KK

L1L1 R1R1

L2L2 R2R2

::

LKLK RKRK

**Output**

Print the number of ways for Tak to go from Cell 11 to Cell NN, modulo 998244353998244353.

**Sample Input 1**Copy

Copy

5 2

1 1

3 4

**Sample Output 1**Copy

Copy

4

The set SS is the union of the segment [1,1][1,1] and the segment [3,4][3,4], therefore S={1,3,4}S={1,3,4} holds.

There are 44 possible ways to get to Cell 55:

* 1→2→3→4→51→2→3→4→5,
* 1→2→51→2→5,
* 1→4→51→4→5 and
* 1→51→5.

**Sample Input 2**Copy

Copy

5 2

3 3

5 5

**Sample Output 2**Copy

Copy

0

Because S={3,5}S={3,5} holds, you cannot reach to Cell 55. Print 00.

**Sample Input 3**Copy

Copy

5 1

1 2

**Sample Output 3**Copy

Copy

5

**Sample Input 4**Copy

Copy

60 3

5 8

1 3

10 15

**Sample Output 4**Copy

Copy

221823067

Note that you have to print the answer modulo 998244353998244353.

1. #include<bits/stdc++.h>
2. #define pb push\_back
3. #define pii pair<int,int>
4. #define int long long int
5. #define vec vector<int>
6. #define mp map<int,int>
7. #define inf 1e18
8. using namespace std;
9. vector<int> v;
10. int mod=998244353;
11. int32\_t main()
12. {
13. ios\_base::sync\_with\_stdio(false);
14. cin.tie(NULL);
15. cout.tie(NULL);
16. int tt=1;
17. //cin>>tt;
18. while(tt--)
19. {
20. int n,k;
21. cin>>n>>k;
22. int l[n],r[n],i;
23. for(i=0;i<k;i++)
24. cin>>l[i]>>r[i];
25. int a[n+1]={0};
26. for(i=0;i<k;i++)
27. {
28. for(int j=l[i];j<=r[i];j++)
29. if(!a[j])
30. {
31. a[j]++;
32. v.pb(j);
33. }
34. }
35. int x=v.size();
36. sort(v.begin(),v.end());
37. int dp[n+1]={0};
38. dp[1]=1;
39. for(i=2;i<=n;i++)
40. {
41. dp[i] = ((dp[i]%mod) + (dp[i-1]%mod))%mod;
42. for(int j=0;j<k;j++)
43. {
44. dp[i] = ((dp[i]%mod) + ((dp[max(i-l[j],(int)0)]%mod - dp[max(i-r[j]-1,(int)0)]%mod + mod)%mod)%mod)%mod;
45. }
46. }
47. cout<<(dp[n]-dp[n-1]+mod)%mod<<"\n";
48. }
49. }

## D - Leaping Tak Editorial by [evima](https://atcoder.jp/users/evima)

First, the naivest solution is a Dynamic Programming (DP) where fifi := the number of ways to go to Cell ii, and check all the possible move for each ii, but this needs a time complexity of O(N2)O(N2).

Let us consider optimizing by making use of the fact that the ways of moving can be written as a sum of small number of segments (S=∪[lj,rj]S=∪[lj,rj]). This can be solved either in so-called "distributing DP" or "receiving DP." This time, we will explain based on distributing DP.

Assume that we have already obtained the values until fifi. When transiting from ii, for all d∈Sd∈S, we want to add fifi to fi+dfi+d.

Here, we want to use the fact that the difference of neighboring values of ff changes at O(K)O(K) positions. Let ai=fi−fi−1ai=fi−fi−1, then this can be optimized by performing the operations of ai+lj:=ai+lj+fi,ai+rj+1:=ai+rj+1−fiai+lj:=ai+lj+fi,ai+rj+1:=ai+rj+1−fi for each jj, and then restore ff by fi=fi−1+aifi=fi−1+ai.

The total time complexity is O(KN)O(KN).

Note that it is known that this problem can be solved in a total of O(NlogN)O(Nlog⁡N) time even when the transitions cannot be written as a sum of small number of segments.

[Sammple Code (C++)](https://atcoder.jp/contests/abc179/submissions/16844205)

posted: about 4 hours ago  
last update: about 4 hours ago

1. #include <bits/stdc++.h>
2. using namespace std;
4. using ll = long long;
5. using pii = pair<int, int>;
6. template <class T>
7. using V = vector<T>;
8. template <class T>
9. using VV = V<V<T>>;
11. #define pb push\_back
12. #define eb emplace\_back
13. #define mp make\_pair
14. #define fi first
15. #define se second
16. #define rep(i, n) rep2(i, 0, n)
17. #define rep2(i, m, n) for (int i = m; i < (n); i++)
18. #define per(i, b) per2(i, 0, b)
19. #define per2(i, a, b) for (int i = int(b) - 1; i >= int(a); i--)
20. #define ALL(c) (c).begin(), (c).end()
21. #define SZ(x) ((int)(x).size())
23. constexpr ll TEN(int n) { return (n == 0) ? 1 : 10 \* TEN(n - 1); }
25. template <class T, class U>
26. void chmin(T& t, const U& u) {
27. if (t > u) t = u;
28. }
29. template <class T, class U>
30. void chmax(T& t, const U& u) {
31. if (t < u) t = u;
32. }
34. template <class T, class U>
35. ostream& operator<<(ostream& os, const pair<T, U>& p) {
36. os << "(" << p.first << "," << p.second << ")";
37. return os;
38. }
40. template <class T>
41. ostream& operator<<(ostream& os, const vector<T>& v) {
42. os << "{";
43. rep(i, v.size()) {
44. if (i) os << ",";
45. os << v[i];
46. }
47. os << "}";
48. return os;
49. }
51. #ifdef LOCAL
52. void debug\_out() { cerr << endl; }
53. template <typename Head, typename... Tail>
54. void debug\_out(Head H, Tail... T) {
55. cerr << " " << H;
56. debug\_out(T...);
57. }
58. #define debug(...) \
59. cerr << \_\_LINE\_\_ << " [" << #\_\_VA\_ARGS\_\_ << "]:", debug\_out(\_\_VA\_ARGS\_\_)
60. #define dump(x) cerr << \_\_LINE\_\_ << " " << #x << " = " << (x) << endl
61. #else
62. #define debug(...) (void(0))
63. #define dump(x) (void(0))
64. #endif
66. template <unsigned int MOD>
67. struct ModInt {
68. using uint = unsigned int;
69. using ull = unsigned long long;
70. using M = ModInt;
72. uint v;
74. ModInt(ll \_v = 0) { set\_norm(\_v % MOD + MOD); }
75. M& set\_norm(uint \_v) { //[0, MOD \* 2)->[0, MOD)
76. v = (\_v < MOD) ? \_v : \_v - MOD;
77. return \*this;
78. }
80. explicit operator bool() const { return v != 0; }
81. M operator+(const M& a) const { return M().set\_norm(v + a.v); }
82. M operator-(const M& a) const { return M().set\_norm(v + MOD - a.v); }
83. M operator\*(const M& a) const { return M().set\_norm(ull(v) \* a.v % MOD); }
84. M operator/(const M& a) const { return \*this \* a.inv(); }
85. M& operator+=(const M& a) { return \*this = \*this + a; }
86. M& operator-=(const M& a) { return \*this = \*this - a; }
87. M& operator\*=(const M& a) { return \*this = \*this \* a; }
88. M& operator/=(const M& a) { return \*this = \*this / a; }
89. M operator-() const { return M() - \*this; }
90. M& operator++(int) { return \*this = \*this + 1; }
91. M& operator--(int) { return \*this = \*this - 1; }
93. M pow(ll n) const {
94. if (n < 0) return inv().pow(-n);
95. M x = \*this, res = 1;
96. while (n) {
97. if (n & 1) res \*= x;
98. x \*= x;
99. n >>= 1;
100. }
101. return res;
102. }
104. M inv() const {
105. ll a = v, b = MOD, p = 1, q = 0, t;
106. while (b != 0) {
107. t = a / b;
108. swap(a -= t \* b, b);
109. swap(p -= t \* q, q);
110. }
111. return M(p);
112. }
114. bool operator==(const M& a) const { return v == a.v; }
115. bool operator!=(const M& a) const { return v != a.v; }
116. friend ostream& operator<<(ostream& os, const M& a) { return os << a.v; }
117. static uint get\_mod() { return MOD; }
118. };
120. using Mint = ModInt<998244353>;
122. int main() {
123. int N, K;
124. cin >> N >> K;
125. V<pii> rng;
127. rep(i, K) {
128. int l, r;
129. cin >> l >> r;
130. r++;
131. rng.eb(l, r);
132. }
134. V<Mint> dp(N);
135. dp[0] = 1;
136. dp[1] = -1;
138. rep(i, N) {
139. if (i > 0) dp[i] += dp[i - 1];
140. rep(j, K) {
141. int l, r;
142. tie(l, r) = rng[j];
143. if (i + l < N) dp[i + l] += dp[i];
144. if (i + r < N) dp[i + r] -= dp[i];
145. }
146. }