More optimized:

|  |
| --- |
| #include <algorithm> |
|  | #include <iostream> |
|  | #include <math.h> |
|  | #include <string.h> |
|  | using namespace std; |
|  |  |
|  | int i, j, TC, xsize, ysize, n, x[11], y[11], dist[11][11], memo[11][1 << 11]; // Karel + max 10 beepers |
|  |  |
|  | int tsp(int pos, int bitmask) { // bitmask stores the visited coordinates |
|  | if (bitmask == (1 << (n + 1)) - 1) |
|  | return dist[pos][0]; // return trip to close the loop |
|  | if (memo[pos][bitmask] != -1) |
|  | return memo[pos][bitmask]; |
|  |  |
|  | int ans = 2000000000; |
|  | for (int nxt = 0; nxt <= n; nxt++) // O(n) here |
|  | if (nxt != pos && !(bitmask & (1 << nxt))) // if coordinate nxt is not visited yet |
|  | ans = min(ans, dist[pos][nxt] + tsp(nxt, bitmask | (1 << nxt))); |
|  | return memo[pos][bitmask] = ans; |
|  | } |
|  |  |
|  | int main() { |
|  | scanf("%d", &TC); |
|  | while (TC--) { |
|  | scanf("%d %d", &xsize, &ysize); // these two values are not used |
|  | scanf("%d %d", &x[0], &y[0]); |
|  | scanf("%d", &n); |
|  | for (i = 1; i <= n; i++) // karel's position is at index 0 |
|  | scanf("%d %d", &x[i], &y[i]); |
|  |  |
|  | for (i = 0; i <= n; i++) // build distance table |
|  | for (j = 0; j <= n; j++) |
|  | dist[i][j] = abs(x[i] - x[j]) + abs(y[i] - y[j]); // Manhattan distance |
|  |  |
|  | memset(memo, -1, sizeof memo); |
|  | printf("The shortest path has length %d\n", tsp(0, 1)); // DP-TSP |
|  | } |
|  |  |
|  | return 0; |
|  | } |