Random walking

Vladimir Demin

October 27, 2009

1. Random walking

1.1 My program

Listing 1: My program code

```
1  #include <stdio.h>
2  #include <time.h>
3  #include <stdlib.h>
4
5  #define POINTS 10000
6  #define LIFES  1000000
7
8  #define TRUE  1
9  #define FALSE 0
10
11 #define TOP    1
12 #define BOTTOM 2
13 #define LEFT   3
14 #define RIGHT  4
15
16 int move() {
17     return (rand() % 4 + 1);
18 }
19
20 int main() {
21
22     int roboX, roboY;
23         int i, k;
24     int shag, die = 0, succecc = 0, age;
25         int number_of_succecc;
26         FILE *out, *all_out, *stat;
27
28     srand(time(NULL));
29
30
31     out = fopen("results.txt", "w");
32     all_out = fopen("all_result.txt", "w");
33     stat = fopen("statistic.txt", "w");
34
35     for (k = 3; k < 101; k++){
36
37       fprintf(out, "==========\n");
38       fprintf(out, "M = %d; N = %d\n", k, k);
39       number_of_succecc = 0;
40
41       for (i = 0; i < LIFES; i++) {
42
43         age = 0;
44         die = FALSE;
45         succecc = FALSE;
46         roboX = 1;
47         roboY = 1;
48
49         do {
50             shag = move();
51             fprintf(out, "%d, ", shag);
52
53             switch (shag) {
```

```
54              case TOP:
55                  if (roboY < k)
56                      roboY++;
57                  break;
58
59              case BOTTOM:
60                  if (roboY > 1) {
61                      roboY--;
62                      if (roboY == 1) {
63                          if (roboX > 1 && roboX < k) {
64                              die = TRUE;
65                          }else if (roboX == k)
66                              succecc = TRUE;
67                      }
68                  }
69                  break;
70
71              case LEFT:
72                  if (roboX > 1)
73                      roboX--;
74                  break;
75
76              case RIGHT:
77                  if (roboX < k) {
78                      roboX++;
79                      if (roboY == 1)
80                          die = TRUE;
81                  }
82                  break;
83
84              default:
85                  break;
86              }
87
88              age++;
89              if (age > POINTS)
90                  die = TRUE;
91          } while (die == FALSE && succecc == FALSE);
92
93          if (succecc == TRUE && die == FALSE) {
94              number_of_succecc++;
95              fprintf(out, "␣-␣succecc\n");
96          }
97          else
98              fprintf(out, "\n");
99        }
100
101       fprintf(out, "\n");
102       fprintf(out, "succecced:␣%d\n", number_of_succecc);
103       fprintf(out, "\n");
104       fprintf(all_out, "%d,␣", k);
105       fprintf(all_out, "%d\n", number_of_succecc);
106       fprintf(stat, "%d␣%d\n", k, number_of_succecc);
107     }
108
109     fclose(all_out);
110     fclose(out);
111
```

```
112     return 0;
113 }
```

With this program I make some tests. Results of this tests may intresting number of succed lifes. Some of this results I put on this report with graphics.

For the first test number of the POINTS have 1001 and number of the LIFES take 10000 (picture 1.1):

N - number of length of the gridworld (N x N) S - number of success lifes

For 51 to 100 lifes number of success was zero. I make graphic with this data:

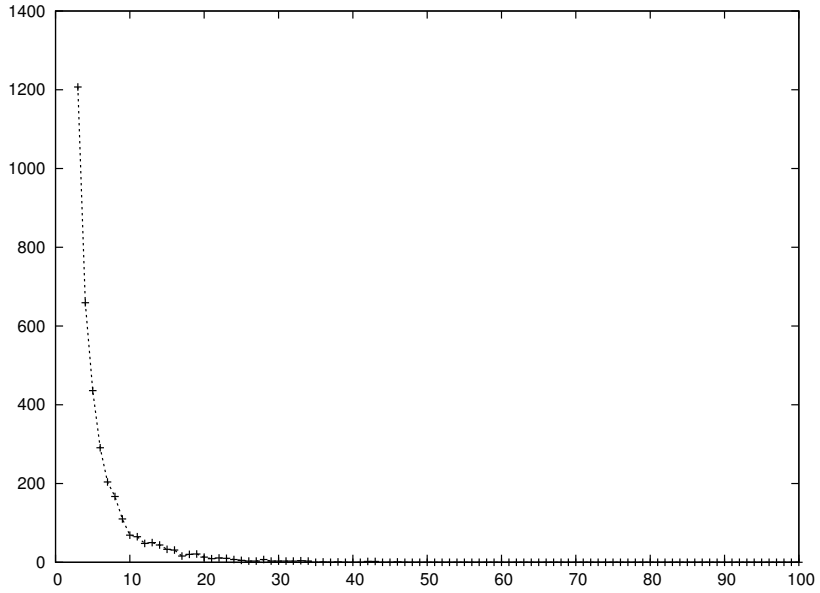

Figure 1: Capture 1.1

On second test I change number of the LIFES to 100000. For this situation I take same graphic (Capture 1.2).

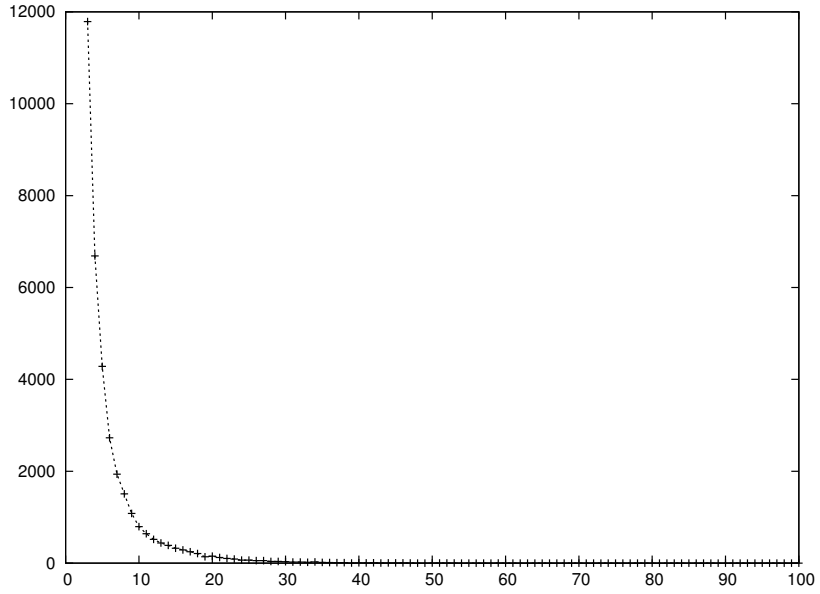

Figure 2: Capture 1.2

Becouse I need 1 successful life for gridworld 100x100 to print it, I change number of the LIFES to 1000000 and number of the POINTS to 10000. Graphic for this test on Capture 1.3.

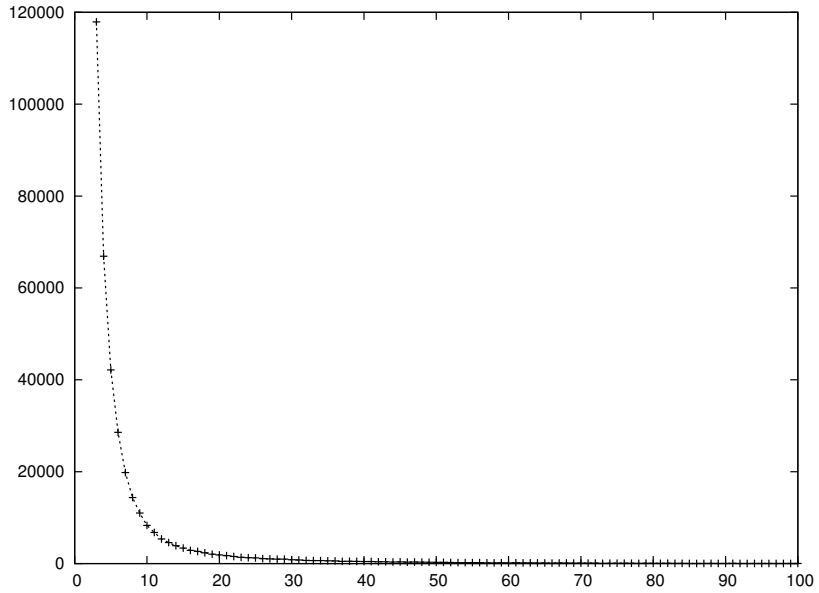

Figure 3: Capture 1.3

Now, I want to show my result for gridworld 30x30.
Next picture is succecc life of robot. He start at (1, 1) and come to (1, 30) (capture 1.7).

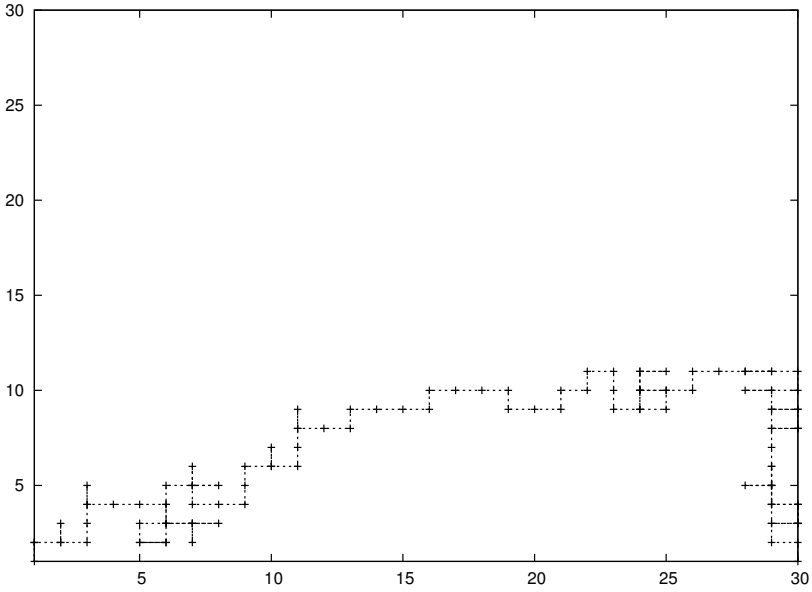

Figure 4: Capture 1.4

In next picture I show way, where robot die on the cliff (capture 1.5):
At last picture robot die, becouse number of steps was more then 150 (capture 1.6):

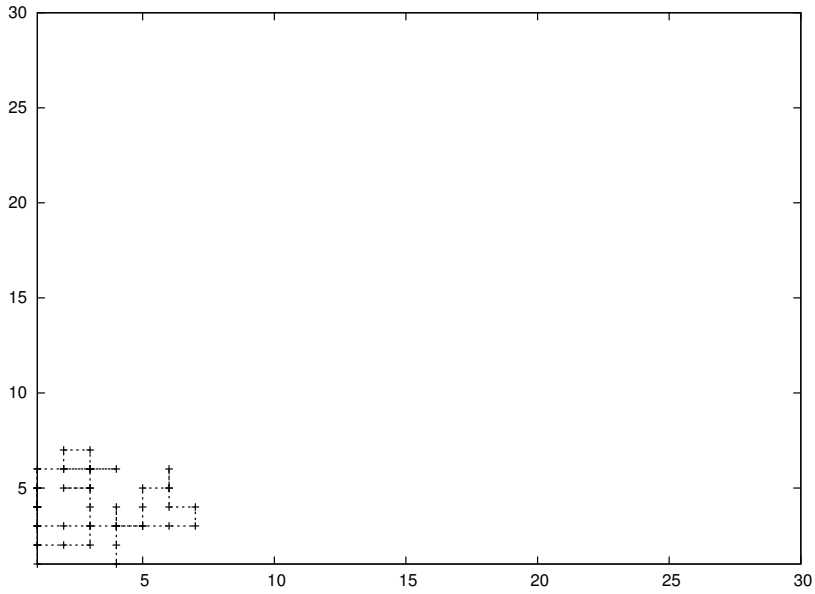

Figure 5: Capture 1.5

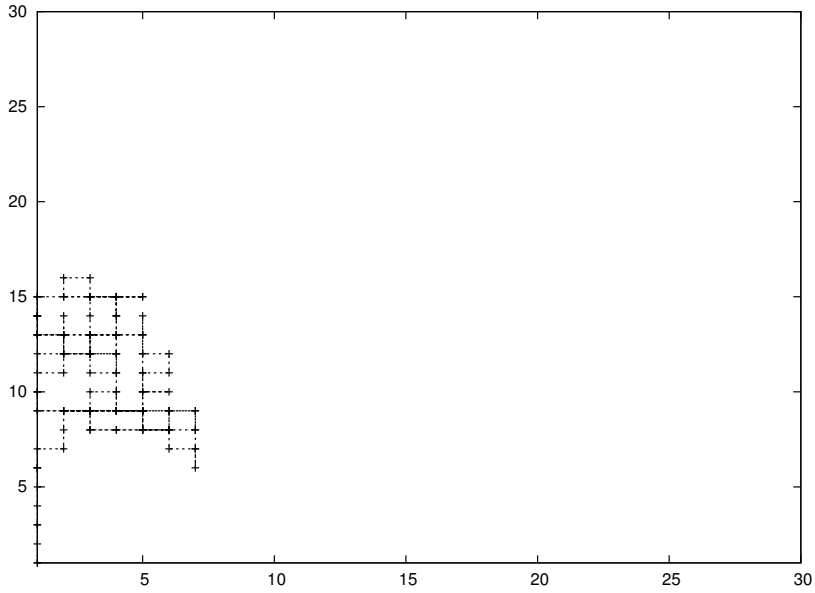

Figure 6: Capture 1.6