第 4 章

「ダイクストラ法とアテナの本」 by 2lu3 (73)

73 回生の 2lu3 です。僕は中 1 までロボカップジュニアというロボコンで自律型ロボットにサッカーをさせる競技をしてきたのですが、中 2 からバーチャルでレスキュー競技を行う CoSpace という種目に転向しました。サッカーとレスキューでは考え方もプログラミングも大分違っている上、C 言語の基礎勉強も同時並行で進める必要があり、何より大会までに時間が全然足りなかったので、とりあえず取り組む課題の中で比較的理解出来そうなことを 1 つ勉強し、それを CoSpace のプログラムに使ってみることにしました。それがダイクストラ法です。試行錯誤してプログラムが出来上がったのは大会直前で、完成度は低かったですが、僕なりに手ごたえを感じることが出来ました。

4.1 CoSpace について

ロボカップジュニアとは

１９才以下のジュニアでチームを組み、自律型ロボット (予めロボットにどう動くかプログラムしてあるロボット↔リモコンロボット) を使って、サッカー・レスキュー・ダンスをそれぞれ競う大会。小学生も大勢参加していて (中高生も多数)、勝ち上がると世界大会まであります。昨年度までずっと国際科学技術コンテスト (数学オリンピック等) の一つでしたが、今年は除外されています。日本ではマイナーで知っている人も少ない大会ですが、海外ではこのジュニアの大会に王族や首相など国のトップが応援に駆け付けたり、良い成績を取れたら大学の特待生になれたり、兵役が免除されたり (命がけ)、国を挙げてサポートしているところもあります。

- ロボカップジュニアア国大会の紹介動画: https://www.youtube.com/watch?v=YchLBl1Vmmw
- ロボカップジュニア HP: http://www.robocupjunior.jp/

CoSpace とは

CoSpace というのはロボカップジュニアのレスキュー競技の中の一種目で、レスキューには中級者向けのライン、上級者向けのメイズ、バーチャルレスキューの CoSpace があります。(レス

キュー競技ではロボットに被災者に見立てたものを回収させゴールまで運ばせて、時間と点数で競います。)

CoSpace はバーチャルの競技なので、ロボコンというよりゲームのように見えると思います。ですが、他の競技と同様に画面上のロボットにはラインセンサー、超音波センサー、方位センサー等が付いていて、スタート地点から発車して被災者を保護し、保護した被災者が 6 人になるとゴール地点に行って降ろします。それを繰り返し、高得点を狙うという競技です。仕掛けも色々あって、点数が 2 倍になるゾーンやトラップ (罠)、沼地、境界線などがあります。また、この被災地マップは大会ごとに異なり、マップ配布後から 3 時間、そのマップでうまく被災者救助できるようにプログラミングします。

- CoSpace の情報発信サイト: http://cospacerobot.org/

4.2 ダイクストラ法について

ダイクストラ法とは

ダイクストラ法といわれても何が何だかわからない方もいると思います。 僕もそうでした。最も説明しやすいのは、ナビです。例えば、あなたの家から駅まで行くには (駅直結タワーマンションとかをのぞき) 普通は必ず道と交差点をいくつか通らないといけません。そして、たくさんある交差点の中から最短でつくことのできるルートを割り出すのがダイクストラ法です。

要領を得ない説明になりましたが、要するに、これは最短ルートを割り出すものってことがわかれば大丈夫です。そしてつまり、これを使えば、CoSpace で障害物 (トラップや沼地など) をよけて被災者を効率良く保護することが可能になります。ダイクストラ法の詳細は、ここのサイトを参考にしました。

- http://www.deqnotes.net/acmicpc/dijkstra/

ソースコード

理論が分かっても、実際にできないと意味がありません。C 言語で書いてみました。

List 4.1 main()

```
1  #include <stdio.h>
2  #include <math.h>
3  #include <string.h>
4
5  #define true 1
6  #define false 0
7  #define NUMBER_OF_NODES 7 //実際の数 + 2
8  #define DONE_NODE NUMBER_OF_NODES-1
9  #define START_NODE 0
10 #define GOAL_NODE 5
11
12 int route_record[NUMBER_OF_NODES];
13 struct Node {
14  int edges_to[NUMBER_OF_NODES];
15  float edges_cost[NUMBER_OF_NODES];
16  int done;
17  float cost;
18  int x;//x座標
19  int y;//y座標
20  int edges_number;//伸びているエッジの数
```

```
};

struct Node nodes[NUMBER_OF_NODES];
  void SettingNode() {
    for (int i = 0; i < NUMBER_OF_NODES; i++) {
    nodes[i].done = 0;
    nodes[i].cost = -1;
    route_record[i] = -1;
    }
  nodes[START_NODE].cost = 0;

  nodes[0].edges_to[0] = 1;
  nodes[0].edges_to[1] = 3;
  nodes[0].x = 0;
  nodes[0].y = 0;
  nodes[0].edges_number = 2;

  nodes[1].edges_to[0] = 0;
  nodes[1].edges_to[1] = 2;
  nodes[1].edges_to[2] = 5;
  nodes[1].x = 20;
  nodes[1].y = 10;
  nodes[1].edges_number = 3;

  nodes[2].edges_to[0] = 1;
  nodes[2].edges_to[1] = 3;
  nodes[2].edges_to[2] = 4;
  nodes[2].x = 20;
  nodes[2].y = 0;
  nodes[2].edges_number = 3;

  nodes[3].edges_to[0] = 0;
  nodes[3].edges_to[1] = 2;
  nodes[3].edges_to[2] = 4;
  nodes[3].x = 10;
  nodes[3].y = -10;
  nodes[3].edges_number = 3;

  nodes[4].edges_to[0] = 2;
  nodes[4].edges_to[1] = 3;
  nodes[4].edges_to[2] = 5;
  nodes[4].x = 30;
  nodes[4].y = -10;
  nodes[4].edges_number = 3;

  nodes[5].edges_to[0] = 1;
  nodes[5].edges_to[1] = 4;
  nodes[5].x = 40;
  nodes[5].y = 0;
  nodes[5].edges_number = 2;

  for (int i = 0; i < NUMBER_OF_NODES; i++) {
      for (int j = 0; j < nodes[i].edges_number; j = j + 1) {
          int a;
          int b;
          a = nodes[i].x - nodes[nodes[i].edges_to[j]].x;
          b = nodes[i].y - nodes[nodes[i].edges_to[j]].y;
          if (a < 0) {
              a = -a;
          }
          if (b < 0) {
              b = -b;
          }
          float c = a*a + b*b;
          c = sqrt(c);
          nodes[i].edges_cost[j] = c;
      }
  }
}

void SearchTheShortestRoad() {
  while (true) {
    nodes[DONE_NODE].cost = -1;
    nodes[DONE_NODE].done = 0;
    for (int i = 0; i < DONE_NODE; i++) {
      if (nodes[i].done == 1 || nodes[i].cost < 0) {
        continue;
      }
      else {
```

```
100         if (nodes[DONE_NODE].done == 0
101             || nodes[i].cost < nodes[DONE_NODE].cost) {
102           nodes[i].done = 1;
103           nodes[DONE_NODE] = nodes[i];
104         }
105       }
106     }
107     if (nodes[DONE_NODE].done == 0) {
108       printf("Sucssefuly\n");
109       break;
110     }
111     nodes[DONE_NODE].done = 1;
112     for (int i = 0; i < nodes[DONE_NODE].edges_number; i++) {
113       int to;
114       float cost;
115       to = nodes[DONE_NODE].edges_to[i];
116       cost = nodes[DONE_NODE].cost + nodes[DONE_NODE].edges_cost[i];
117       if (nodes[to].cost < 0 || cost < nodes[to].cost) {
118           nodes[to].cost = cost;
119       }
120     }
121   }
122 }
123
124 void InvestigaterouteRecord() {//経路を調べる
125   int checked_node = GOAL_NODE;//経路がわかっていて出発点に最も近いノード
126   route_record[0] = -1;
127   int checking_node;
128   int i;
129   while (true) {
130     for (i = 0; i < nodes[checked_node].edges_number; i++) {
131       checking_node = nodes[checked_node].edges_to[i];
132       //経路に含まれるか現在調べているノード
133       if (nodes[checked_node].cost < nodes[checked_node].edges_cost[i]
134           + nodes[checking_node].cost + 5
135           &&
136           nodes[checked_node].cost > nodes[checked_node].edges_cost[i]
137           + nodes[checking_node].cost - 5
138           ) {
139           //現在ノードのコストは
140           //それら2つをつなぐエッジのコスト+試しているノードのコスト
141             break;
142       }
143     }
144     for (i = 0; i < NUMBER_OF_NODES; i++) {
145       if (route_record[i] == -1) {
146         route_record[i] = checking_node;
147         i = i + 1;
148         route_record[i] = -1;
149         break;
150       }
151     }
152     checked_node = checking_node;
153     if (checked_node == START_NODE) {
154       break;
155     }
156   }
157 }
158
159 int main() {
160   SettingNode();
161   SearchTheShortestRoad();
162   InvestigaterouteRecord();
163   printf("cost = %d \n", nodes[GOAL_NODE].cost);
164   printf(
165   "注意 逆順です。-1は余った分なので気にしないでください\n%d %d %d %d %d\n"
166   , route_record[0], route_record[1], route_record[2],
167   route_record[3], route_record[4]);
168   return 0;
169 }
```

以上です。

4.3 『 アテナの本 』構想の紹介

小中学生の君へ

小中学生の皆さん、好きなことや得意なことってありますか? 「ある!」と即答できる人はとてもラッキーです。このままそれをどんどん続けていって下さい。答えるのに少し考えた人や、「ない(かも)」と思った人は、そういうものをこれから見つけた方が毎日楽しくなります。でも、何をしたらいいのか、何が得意になるのか、見つけ方も分からないという子がいるかもしれません。ロボットやプログラミングを始める前は僕もそうでした。

好きなこと得意になるものというのは、純粋に君自身がやってみて「楽しい!」と強く思ったものです。そして、「ものすごく楽しい!」と思ったことは、これから君を伸ばしてくれるジェット燃料になります。楽しい何かを目一杯楽しむことで、君の学力はもちろん、色々な力を伸ばしてくれるからです。そして、君が好きなことを頑張ることで、君の周りでも頑張り始める子が出てくるだろうし、そのまた周囲も頑張るだろうし、そんな感じで君が大きくなっていけば、やがて日本がほんの少し良くなっていく、かもしれません。

例えば、TV で見た人も多いでしょうが、月でロボットに５００mのかけっこをさせるという国際レースに、日本からただ 1 チームだけ、HAKUTO というチームが参加しています。この大会は優勝賞金が 2000 万ドル (約 20 億円) なのですが、ロボットを作る技術や費用も、ロボットを月に飛ばす費用も大変すぎるので、とてつもない挑戦です。それでも、どうしても挑戦してみたい!と思った袴田さんというたった一人の夢への情熱が、大勢の人達の心を動かし (大勢の専門家がボランティアとして集まり)、大企業も動かし (スポンサーになってロボット開発費を出してくれて)、荒唐無稽とも思われていた夢を実際に実現に向け、王手をかけつつあります。(世界中から 18 チーム参加しているのですが、HAKUTO は昨年中間賞を取り、今年最終選抜の 5 チームに入りました。そして、今年 12 月末、いよいよアメリカで打ち上げです!) 最初に知った時にはこんな面白いことに挑戦してる日本人がいるということにとても感激したし、とても勇気づけられました。TV で見た人も大勢感激しただろうと思います。

- HAKUTO の HP: https://team-hakuto.jp/index.html

話を元に戻します。そういう不思議な力のある、君がこれから好きになることというのが、身近に知っていたり取り組んでいる人がいるようならとても理想的ですが、科学館や博物館、街中でも見つかるかもしれません。それに、今日の文化祭もかなりチャンスと思います。ここパソコン研究部はじめ、面白すぎる企画が盛りだくさんだし、どこでも中高生がいれば質問も気軽にできます。君が思い切って質問してみたら、ちゃんと答えてくれると思います。

文化祭のように実際に見たり聞いたりした方が良いのですが、もっと気軽に、好きなことを見つけるきっかけの一つになればと思って、個人的な趣味で『アテナの本』という HP を作ってみました。

- アテナの本: https://book4athena.amebaownd.com/

まだコンテンツはほんの少しで、これから少しずつ増やしていきたいと思っている構想段階なのですが、よかったらのぞいてみて下さい。(※ご注意: この HP はパソコン研究部や学校とは全く無関係です。)

アテナの本構想とは

これから協力者 (中高生ボランティア) を増やして、「得意なことのある中高生と学びたい小中学生をつなぎ、小中学生がこれから得意になることをみつける最初の一歩を手伝う、知的プラットフォーム」となることを目指しています。

具体的な内容は以下の通りです。

1. 中高生: 中高生ボランティアが自分の取り組んでいることを小学生に分かるようにまとめ (数分程度の動画メイン)、それを HP に掲載します。
2. 小中学生: 好きなページを自由に見て回って、中高生の取り組みを学びます。
3. 小中学生: 「やってみたい!」と思ったら、最初の一歩は具体的にどうやったのか体験談も紹介してあるのでそれを参考にして、自分でもやってみよう!
4. 小中学生: 質問もできます。[*1]
5. 中高生: 時間がかかるかもしれませんが、HP 上でまとめて回答していきます。
6. 小中学生: コンテストや大会に出場してみよう♪
7. 大会に出場したかつての小中学生: 中高生ボランティアとなって下さい!
8. 中高生: 企画力・プレゼン力・対応力が身に付き、社会貢献でき、学校を超えた仲間も出来ます。大学入試改革 (学業以外も少し考慮という方向?) で、広くたくさんのボランティアが集まるといいなあと思っています。

これらは構想段階で、今はお試し運用中という状況です。考査やイベントを除く期間にマイペースとなりますが地道に運営していきますので、温かい目で見守ってもらえたら嬉しく思います。よろしくお願い致します。

[*1] 最初の一歩を踏み出す応援なので、最初の段階での質問となります。ご理解下さい。