Treball pràctic individual: Programació d'un algoritme A^*

Marc Roig Oliva
1668300

Índex

1 Introducció

Es requereix la programació d'una utilitat que trobi el cami més ràpid entre dos punts utilitzant l' algorisme de cerca $A*$.

El funcionament de $A*$ consisteix en una funció d'avaluació que representa el pes de cada node: $f(n) = g(n) + h'(n)$ on $f(n)$ és el pes total, $g(n)$ es el cost per arribar des de l'orígen fins a un determinat node i $h'(n)$ representa el cost per arribar des del node actual fins al destí a priori. A més s'implementa una cua ordenada pel pes total (el valor de $f(n)$) de cada node.[1]

El lliurament tracta d'un programa escrit en C que implementa l'agorisme de cerca, la lectura de les dades dels fitxers i l'emmagatzematge de aquestes, implementacions de cues i una rutina que mostra el camí recorregut i la distància total.

El programa s'ajuda d'un fitxer de carrers extret de `openstreetmaps` i d'un fitxer de nodes creat a partir de l'anterior, s'asumeix que tots els camins son bidireccionals i que el fitxer de nodes conté els nodes de forma ordenada.

2 Execució

El programa ha estat compilat i executat a una distribució de Linux fent us del compilador `gcc` de la següent manera:

```
$ gcc AEstrella.c -o AEstrella -lm -Wall
```

```
$ ./AEstrella <node_origen> <node_desti>
```

La primera sentència genera un executable i la segona (en cas s'haver introduït dos nodes vàlids) mostra el camí mes ràpid entre els dos nodes o en el seu defecte un missatge indicant que no s'ha trobat una ruta possible.

[1] Definició extreta de l'article de Wikipedia sobre $A*$

3 Estructura del codi

3.1 Structs

3.1.1 Gestió de cues

```
typedef struct Element{
    int node;
    struct Element *seg;
}ElementCua;

typedef struct{
    ElementCua *inici;
    ElementCua *final;
}UnaCua;
```

Les estructures de gestió de cues son `ElementCua` i `UnaCua`, la primera conte l'index del node a la llista de nodes i l'element següent, i la segona guarda la posició inicial i final per tal de poder recòrrer la cua.

3.1.2 Gestió de nodes

```
typedef struct{
    char carrer[12];
    int numnode;
    double llargada;
}infoaresta;

typedef struct{
    long int id;
    double latitud, longitud;
    int numarst;
    double pes;
    double dist_origen;
    double dist_desti;
    int anterior;
    infoaresta *arestes;
}Node;
```

En aquest cas l'estructura `infoaresta` guarda la llargada la identificació del carrer i el número del node que conecta, mentre que `Node` guarda tota la informació associada a cada node, com pot ser la identificació del node, les seves coordenades al mapa, el numero de arestes que te associat, el node anterior pel que hem passat i les components de la funció heuristica descrita a la introducció (f ve donada per `pes`, g ve donada per `dist_origen` i h per `dist_desti`).

3.2 Funcions

3.2.1 buscapunt

```
unsigned buscapunt(Node *, int, long int);
```

Donada la llista de nodes, la longitud de la llista i una identificació, retorna la posicio del node amb aquella identificació.

3.2.2 distancia

```
double distancia(Node, Node);
```

Calcula la distància entre dos nodes.

3.2.3 encua

```
void encua(UnaCua *, unsigned);
```

Donat un node, l'afegeix a la cua.

3.2.4 indexoptim

```
int indexoptim(UnaCua *, Node *);
```

Busca l'index de l'element amb $f(n)$ més petit.

3.2.5 desencua

```
void desencua(UnaCua *, Node *);
```

Treu de la cua l'element amb $f(n)$ més petit.

3.2.6 esAlaCua

```
int esAlaCua(UnaCua *, int);
```

Donat un node retorna un booleà especificant si està encuat o no.

3.2.7 mostracami

```
void mostracami(Node * , int, long int, long int , int , int);
```

Imprimeix per consola el camí més curt calculat per A^*.

3.3 Main

En primer lloc el programa valida el número d'arguments introduits i els guarda.

```
if (argc == 3) {
    sscanf(argv[1], "%ld", &IDsortida);
    sscanf(argv[2], "%ld", &IDdesti);
} else {
    printf("El programa necessita 2 arguments i s'han introduit %d", argc-1);
    return -1;
}
```

Posteriorment es procedeix a la lecura dels fitxers de Nodes i Carrers i s'emmagatzemen les dades que contenen[2]

```
nnodes = 0;
while ((c = fgetc(dadesNodes)) != EOF){
    if (c == '\n') nnodes++;
}

Node * punts;

if ((punts= (Node *)malloc(nnodes * sizeof(Node))) == NULL){
    printf("Error malloc nodes\n");
    return 2;
}

rewind(dadesNodes);

for (int i = 0; i < nnodes; i++){
    fscanf(dadesNodes, "%ld;", &punts[i].id);
    fscanf(dadesNodes, "%lf;", &punts[i].latitud);
    fscanf(dadesNodes, "%lf\n;", &punts[i].longitud);
}
fclose(dadesNodes);

if ((dadesCarrers = fopen("Carrers.csv", "r")) == NULL){
        printf("No es possible obrir el fitxer de dades Carrers.csv\n");
        return 1;
}
```

[2]No esta inclosa la part del codi on s'inicialitzen els paràmetres i es reserva la memòria

```
char IDcarrer[12];
long int nodeID;

while ((c=fgetc(dadesCarrers))!=EOF){
    fscanf(dadesCarrers,"d=%[0-9]", IDcarrer);
    fscanf(dadesCarrers,";%ld",&nodeID);
    int posant = buscapunt(punts, nnodes, nodeID);

    while (posant == MAX && fgetc(dadesCarrers) != '\n'){
        printf("%010ld no existeix\n", nodeID);
        fscanf(dadesCarrers,"%ld",&nodeID);
        posant = buscapunt(punts, nnodes, nodeID);
    }

    while (fgetc(dadesCarrers)!='\n'){
        fscanf(dadesCarrers,"%ld",&nodeID);
        int pos = buscapunt(punts, nnodes, nodeID);
        while ((pos == MAX) && (fgetc(dadesCarrers) != '\n')){
            printf("%010ld no existeix\n", nodeID);
            fscanf(dadesCarrers,"%ld",&nodeID);
            pos = buscapunt(punts, nnodes, nodeID);
        }
        if (pos < MAX){
            strcpy(punts[pos].arestes[punts[pos].numarst].carrer, IDcarrer);
            punts[pos].arestes[punts[pos].numarst].numnode = posant;
            punts[pos].numarst++;
            strcpy(punts[posant].arestes[punts[posant].numarst].carrer, IDcarrer);
            punts[posant].arestes[punts[posant].numarst].numnode = pos;
            punts[posant].numarst++;
        }
        posant = pos;
    }
}
fclose(dadesCarrers);
```

Es fa una validació dels nodes d'entrada i sortida per comprovar si son vàlids

```
if (!(IDXsortida = buscapunt(punts, nnodes, IDsortida)))
{
    printf("El node de sortida no es valid\n");
    return -1;
}
if (!(IDXdesti = buscapunt(punts, nnodes, IDdesti)))
{
    printf("El node de desti no es valid\n");
    return -1;
}
```

Per últim es fa la cerca del camí més òptim amb A^* i es mostra el camí.

```
while (nodesCua.inici != NULL){

    IDXactual = indexoptim(&nodesCua, punts);
    if (punts[IDXactual].id == IDdesti){
        break;
    }
    desencua(&nodesCua, punts);
    int next;
    for (int i = 0; i < punts[IDXactual].numarst; i++){
        next = punts[IDXactual].arestes[i].numnode;
        distAux = punts[IDXactual].dist_origen + distancia(punts[IDXactual], punts
    [next]);
        if (distAux < punts[next].dist_origen){
            punts[next].anterior = IDXactual;
            punts[next].dist_origen = distAux;
            punts[next].pes = punts[next].dist_origen + punts[next].dist_desti;

            if (!esAlaCua(&nodesCua, next)) encua(&nodesCua, next);
        }
    }
}
mostracami(punts, IDXactual, IDsortida, IDdesti, IDXsortida, IDXdesti);
```

4 Exemples de funcionament

4.1 Mapa de Sabadell

Orígen:0259184345, Destí:1793441250

```
#La distancia de 259184345 a 1793441250 es de 507.886804 metres
#Cami optim:
Id=0259184345 | 41.545380 | 2.106830 | Dist=0.000000
Id=0259437888 | 41.545752 | 2.106744 | Dist=42.042027
Id=0259437888 | 41.545752 | 2.106744 | Dist=42.042027
Id=0259437905 | 41.546388 | 2.106495 | Dist=115.722401
Id=0259438253 | 41.546734 | 2.107858 | Dist=235.476559
Id=0965459173 | 41.546963 | 2.107746 | Dist=262.617111
Id=0960085142 | 41.547169 | 2.107648 | Dist=286.900381
Id=1944921315 | 41.547281 | 2.107553 | Dist=301.623075
Id=2412854895 | 41.547777 | 2.107092 | Dist=368.879522
Id=1944921533 | 41.548161 | 2.107668 | Dist=433.058493
Id=1944921536 | 41.548240 | 2.107798 | Dist=446.899876
Id=1944921547 | 41.548280 | 2.107864 | Dist=454.047084
Id=1793441253 | 41.548396 | 2.108055 | Dist=474.495887
Id=1944921549 | 41.548399 | 2.108073 | Dist=476.041258
Id=1955175329 | 41.548407 | 2.108110 | Dist=479.200110
Id=1955175330 | 41.548452 | 2.108329 | Dist=498.154671
```

Orígen:0255402782, Destí:0960085144

```
#La distancia de 255402782 a 960085144 es de 1230.835346 metres
#Cami optim:
Id=0255402782 | 41.546260 | 2.111977 | Dist=0.000000
Id=0259190436 | 41.545833 | 2.110622 | Dist=122.272964
Id=0259190436 | 41.545833 | 2.110622 | Dist=122.272964
Id=0255402706 | 41.545478 | 2.109556 | Dist=219.427452
Id=0255401832 | 41.545457 | 2.108940 | Dist=270.697262
Id=0255401552 | 41.545445 | 2.108236 | Dist=329.375343
Id=1945672704 | 41.545470 | 2.108236 | Dist=332.233489
Id=1945672727 | 41.545482 | 2.108236 | Dist=333.479123
Id=1945672745 | 41.545538 | 2.108238 | Dist=339.795921
Id=1945672751 | 41.545575 | 2.108233 | Dist=343.854550
Id=0259437804 | 41.545636 | 2.108224 | Dist=350.696347
Id=1945672833 | 41.545728 | 2.108214 | Dist=361.019637
Id=1945672856 | 41.545899 | 2.108148 | Dist=380.734984
Id=0259437805 | 41.545943 | 2.108129 | Dist=385.921319
Id=4689121780 | 41.546064 | 2.108116 | Dist=399.352830
Id=0961560977 | 41.546482 | 2.107954 | Dist=447.820512
Id=0259438253 | 41.546734 | 2.107858 | Dist=476.923917
Id=0965459173 | 41.546963 | 2.107746 | Dist=504.064469
Id=0960085142 | 41.547169 | 2.107648 | Dist=528.347739
Id=1944921315 | 41.547281 | 2.107553 | Dist=543.070434
Id=2412854895 | 41.547777 | 2.107092 | Dist=610.326880
Id=0960085139 | 41.548042 | 2.106847 | Dist=646.118873
Id=0960085129 | 41.548241 | 2.106649 | Dist=810.436128
Id=0960085143 | 41.548884 | 2.106092 | Dist=895.723268
Id=0960085140 | 41.549498 | 2.105462 | Dist=1037.838288
Id=1446249061 | 41.550181 | 2.104840 | Dist=1129.782685
```

4.2 Mapa d'Andorra

Orígen:9725886266, Destí:9725933859

```
#La distancia de 9725886266 a 9725933859 es de 911.516220 metres
#Cami optim:
Id=9725886266 | 42.510936 | 1.533389 | Dist=0.000000
Id=9725886265 | 42.510919 | 1.533378 | Dist=2.046633
Id=9725886265 | 42.510919 | 1.533378 | Dist=2.046633
Id=9725886264 | 42.510896 | 1.533390 | Dist=4.717324
Id=9725886273 | 42.510766 | 1.533499 | Dist=21.691967
Id=9725886263 | 42.510722 | 1.533536 | Dist=27.489591
Id=3662696505 | 42.510715 | 1.533528 | Dist=28.485100
Id=9725886274 | 42.510594 | 1.533378 | Dist=46.749725
Id=0051410094 | 42.510560 | 1.533336 | Dist=51.868458
Id=7090620267 | 42.510484 | 1.533213 | Dist=65.012139
Id=9725876855 | 42.510494 | 1.533259 | Dist=68.927477
Id=7090620268 | 42.510490 | 1.533303 | Dist=72.554626
Id=7090620265 | 42.510466 | 1.533342 | Dist=76.695542
Id=0265050188 | 42.510300 | 1.533480 | Dist=98.372607
Id=7090684971 | 42.510171 | 1.533602 | Dist=115.876752
Id=7090684970 | 42.510154 | 1.533630 | Dist=118.857183
Id=0051405270 | 42.510125 | 1.533702 | Dist=125.528997
Id=0051405269 | 42.510126 | 1.533742 | Dist=128.776158
Id=0646808608 | 42.510122 | 1.533781 | Dist=132.028040
Id=7090684977 | 42.510105 | 1.533837 | Dist=136.990702
Id=0051405268 | 42.510078 | 1.533885 | Dist=141.946838
Id=0646808603 | 42.510042 | 1.533921 | Dist=146.906940
Id=9726521340 | 42.510000 | 1.533943 | Dist=151.870296
Id=0051409972 | 42.509965 | 1.533947 | Dist=155.824614
Id=0646808597 | 42.509958 | 1.533948 | Dist=156.628609
Id=0051405266 | 42.509915 | 1.533938 | Dist=161.389296
Id=0051405265 | 42.509877 | 1.533914 | Dist=166.154789
Id=0646808596 | 42.509837 | 1.533866 | Dist=172.092794
Id=0051405263 | 42.509811 | 1.533802 | Dist=178.023607
Id=2188366030 | 42.509948 | 1.533727 | Dist=194.531288
Id=2188401567 | 42.510463 | 1.533296 | Dist=261.737320
Id=2188401568 | 42.510733 | 1.533656 | Dist=303.877716
Id=2758586334 | 42.511121 | 1.533433 | Dist=350.713871
Id=2758586336 | 42.511256 | 1.533263 | Dist=371.223169
Id=9725847742 | 42.512238 | 1.532780 | Dist=487.295267
Id=2794827601 | 42.512656 | 1.533339 | Dist=552.582460
Id=9725933964 | 42.512709 | 1.533289 | Dist=559.765371
Id=9725933963 | 42.512639 | 1.533202 | Dist=570.368188
Id=9725933962 | 42.512701 | 1.533108 | Dist=580.700209
Id=9725933961 | 42.512766 | 1.533059 | Dist=588.954067
Id=9725933960 | 42.512703 | 1.532995 | Dist=597.662326
Id=9725933959 | 42.512624 | 1.532913 | Dist=608.761235
Id=9725933958 | 42.512654 | 1.532850 | Dist=614.964358
Id=9725933934 | 42.512642 | 1.532852 | Dist=616.380320
Id=2794827603 | 42.512510 | 1.532537 | Dist=646.092358
Id=2794827599 | 42.512347 | 1.532617 | Dist=665.379381
Id=2188646225 | 42.512285 | 1.532537 | Dist=674.856033
Id=2188646193 | 42.512150 | 1.531938 | Dist=726.204216
Id=9725847743 | 42.511772 | 1.532128 | Dist=771.123081
Id=2794827598 | 42.511500 | 1.531152 | Dist=856.621449
Id=9725847760 | 42.511369 | 1.530764 | Dist=891.660263
Id=9725933857 | 42.511398 | 1.530745 | Dist=895.277497
Id=9725933858 | 42.511434 | 1.530847 | Dist=904.559190
```

5 Representació del camí al mapa

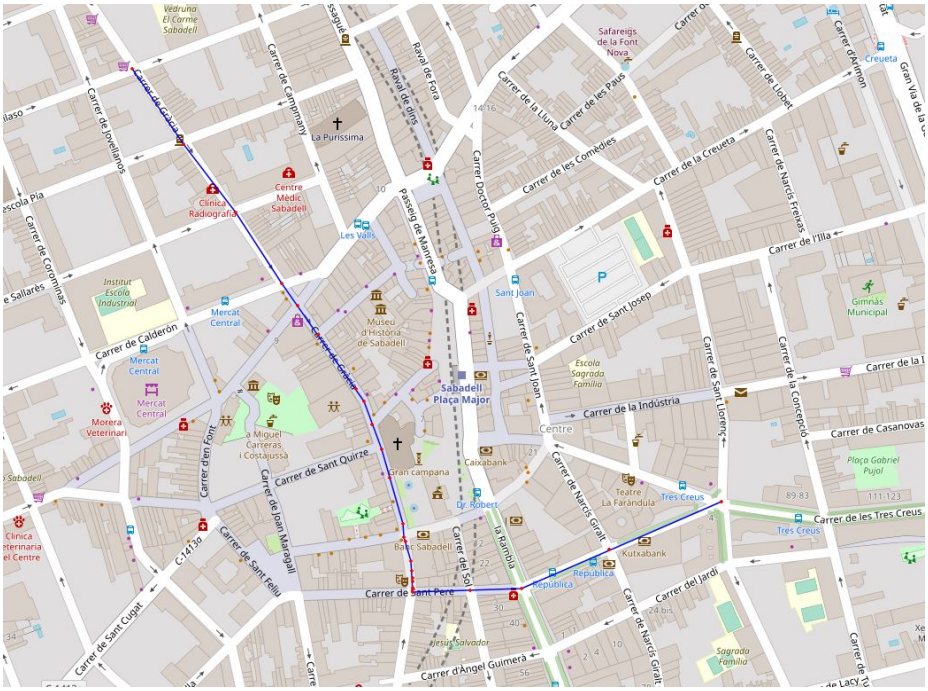

Figure 1: Origen: 0255402782, Destí: 0960085144 - Sabadell

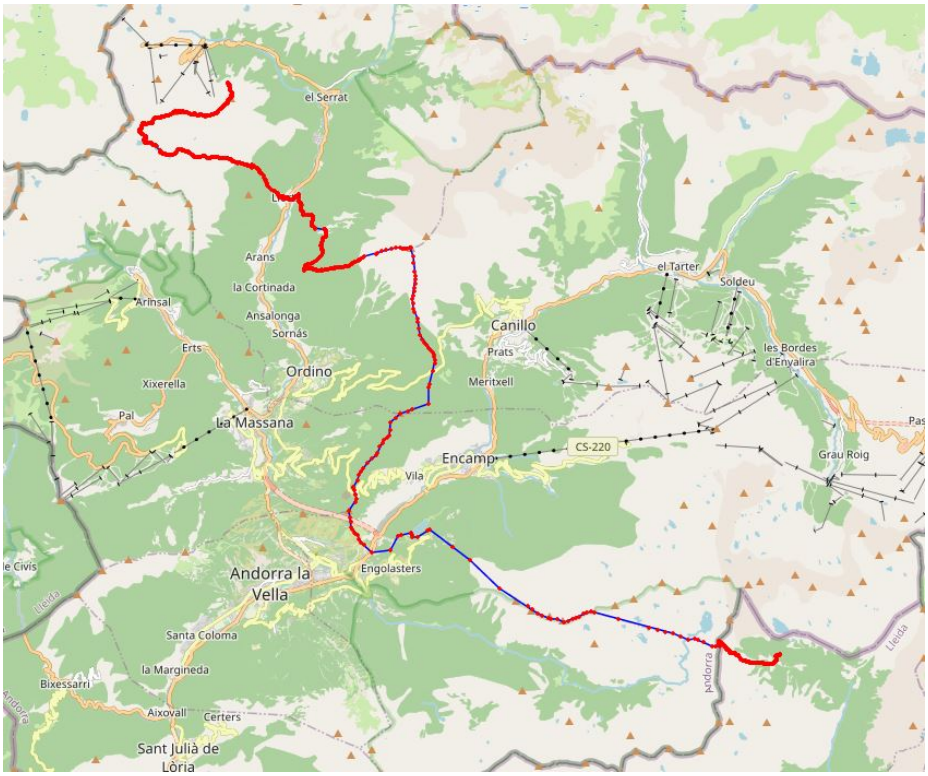

Figure 2: Origen: 0482124451, Destí: 5097996953 - Andorra