Laporaan Tugas Kecil 3

IF2211- Strategi Algoritma

13522130

Justin Aditya Putra Prabakti

Teori dasar

Word Ladder adalah sebuah permainan dimana pemain harus mengubah suatu kata menjadi kata lain dengan syarat hanya boleh satu huruf setiap gantinya. Walaupun memang memerlukan tingkat kemahiran inggris diatas rata-rata untuk mengalahkan permainan ini, namun melalui teknik "brute-force" kita sebenernya bisa mendapatkan jawaban untuk puzzle word ladder (selama kosa katanya bisa berubah.

Melihat kembali adanya "hubungan" antar dua kata yang berbeda satu huruf, permasalahan ini bisa diabstraksikan melalui konsep tree dan diselesaikan menggunakan algoritma bersangkutan. Dan memang ketika dilihat secara sekilas ketiga algoritma ini sama saja. Namun terdapat perbedaan yang cukup jelas. UCS mementingkan cost rendah.

Implementasi

Pada tugas kali ini, kami ditugaskan untuk mengimplementasikan 3 algoritma: Unified Cost Search, Greedy Best First Search, serta A*. Semua algoritma ini mirip, dalam arti tiga hal tersebut dapat digunakan dan diterapkan Ketika menghadapi suatu permasalahan yang dapat abstraksikan sebagai pohon. Namun ketika algoritma tersebut melihat hal yang berbeda.

Algoritma UCS hanya bisa melihat node yang bertetanggaan dengan nya, sehingga akan bulak balik selalu mendahulukan jalur yang paling "Murah" baginya. Untuk tugas ini jalur yang dihitung untuk UCS adalah jumlah kata yang telat dilalui dari kata awal.

Implementasi UCS pada tugas ini :

1. Mulai dari kata awal dan suatu priorityqueue kosong.
2. Masukkan semua tetangga dari posisi sekarang yang belum pernah didatangi kedalam priorityqueue.
3. Pilih node dalam priority queue yang paling pendek.
4. Ulangi 2-3 hinga ditemukan titik tujuan.

Sedangkan untuk GBFS, GBFS melakukan hamper kebalikkan dari UCS, Dimana GBFS hanya melihat titik ahkir dan memilih jalur yang paling meyakinkan. Contohnya pada tugas ini, semakin banyak huruf yang sama dengan huruf tujuan, maka akan semakin diminati oleh GBFS.

Implementasi GBFS pada tugas ini :

1. Mulai dari kata awal dan suatu priorityqueue kosong.
2. Masukkan semua tetangga dari posisi sekarang yang belum pernah didatangi kedalam priorityqueue.
3. Pilih node yang memiliki huruf paling banyak yang sama dengan kata tujuan.
4. Ulangi 2-3 hingga ditemukan titik tujuan.

Terahkir adalah A* (A star). A star merupakan gabungan dari kedua UCS dan BFS Dimana A star akan mengutamakan jalur yang paling pendek, namun juga memperhitungkan jaraknya dari target yang ingin dituju, supaya jalan yang diprioritaskan selalu mengarah ke titik ahkir.

Implementasi A Star pada tugas ini:

1. Mulai dari kata awal dan suatu priorityqueue kosong.
2. Masukkan semua tetangga dari posisi sekarang yang belum pernah didatangi kedalam priorityqueue.

3. Pilih node yang memiliki huruf paling banyak yang sama dengan kata tujuan DAN yang paling pendek.
4. Ulangi 2-3 hingga ditemukan titik tujuan.

Hasil

Test case 1 : SWORD-> WHITE

UCS :

```
FOUND
white
[sword, swore, shore, shire, shine, whine]
UCS :
Nodes visitted : 675
TIme taken : 297 Miliseconds
Memory used : 1810696 Bytes
```

GBFS :

```
FOUND
white
[sword, swore, shore, shire, shine, whine]
GBFS :
Nodes visitted : 9
TIme taken : 15 Miliseconds
Memory used : 41080 Bytes
```

A* :

```
[sword, swore, shore, shire, shine, whine]
A* :
Nodes visitted : 7
TIme taken : 16 Miliseconds
Memory used : 57576 Bytes
```

Test case 2 : LOGIC-> HEART

UCS :

```
FOUND
heart
[logic, yogic, yogis, yagis, ragis, ranis, rands, rends, reads, heads, hears]
UCS :
Nodes visitted : 4829
TIme taken : 1898 Miliseconds
Memory used : 6622112 Bytes
Choose algorithm :
```

GBFS :

```
FOUND
heart
[logic, yogic, yogis, yagis, ragis, rages, reges, rexes, hexes, hemes, hemps, heaps, hears]
GBFS :
Nodes visitted : 49
TIme taken : 30 Miliseconds
Memory used : 82200 Bytes
```

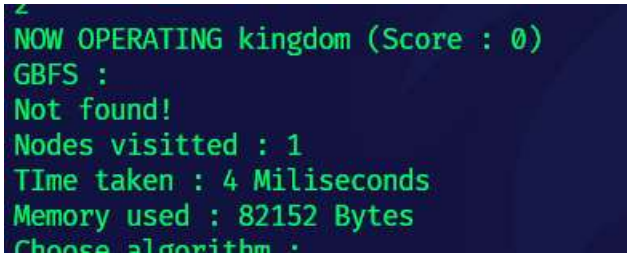
A* :

```
[logic, yogic, yogis, yagis, ragis, ragas, ralas, raids, caids, cains, carns, carrs, carry, harry, herry, berry, beery, beers, bears, hears]
A* :
Nodes visitted : 29
TIme taken : 45 Miliseconds
Memory used : 170336 Bytes
Choose algorithm :
```

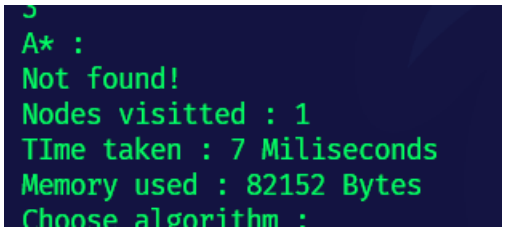
Test case 3 : KINGDOM-> FIGHTER

UCS :

GBFS :

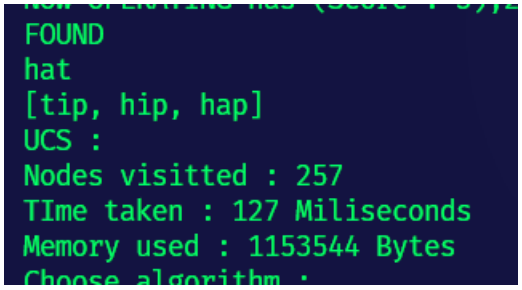


A* :

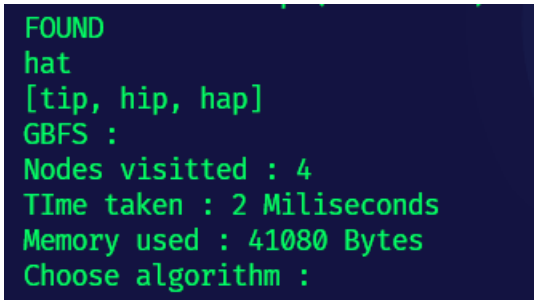


Test case 4 : TIP-> HAT

UCS :



GBFS :



A* :

```
[tip, hip, hap]
A* :
Nodes visitted : 4
TIme taken : 15 Miliseconds
Memory used : 86424 Bytes
```

Test case 5 : TROD -> WALK

UCS :

```
NOW OPERATING wawl (Score : 7),3234
FOUND
walk
[trod, prod, pood, pool, poll, pall, wall]
UCS :
Nodes visitted : 3235
TIme taken : 1111 Miliseconds
Memory used : 4853528 Bytes
```

GBFS :

```
FOUND
walk
[trod, prod, pood, wood, wold, weld, well, wall]
GBFS :
Nodes visitted : 17
TIme taken : 9 Miliseconds
Memory used : 41080 Bytes
Choose algorithm :
```

A* :

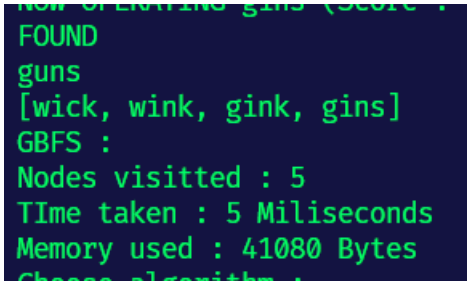
```
ADD, work (19 | 2)
[trod, prod, plod, clod, clad, chad, cham, wham, whap, whip, whin, wain, waif, waff, waft, want, wand, ward, wark]
A* :
Nodes visitted : 22
TIme taken : 41 Miliseconds
Memory used : 187752 Bytes
Choose algorithm :
```

Test case 6 : WICK -> GUNS

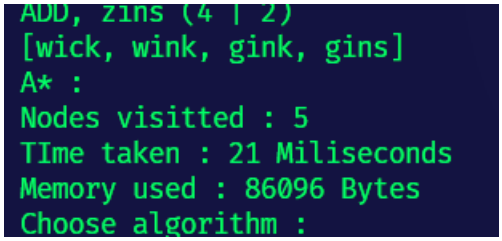
UCS :

```
FOUND
guns
[wick, wink, gink, gins]
UCS :
Nodes visitted : 1265
TIme taken : 431 Miliseconds
Memory used : 2756416 Bytes
```

GBFS :



A* :



Analisis

Secara keseluruhan, pada permasalahan ini algoritma GBFS menghasilkan hasil yang paling memuaskan dan dapat mengalahkan UCS dan A*. Namun, dalam segi pendek nya jawaban, UCS memiliki keunggulan yang lebih diabanding algoritma lainnya, ini dikarenakan perhitungan UCS yang menghitungkan jarak terlebih dahulu, sehingga sudah pasti akan mendapatkan rute terdepat.

Sisi heuristik A-Star kurang memuaskan pada kasus ini karena kita tidak seperti peta dalam dunia 2d yang sudah diketahui keberadaan titik awal, ahkir, dan segala rute/node yang mungkin. Berbeda dengan node kata-kata yang dapat terhubung ke banyak sekali node lainnya.

Lampiran

Github :

https://github.com/BiZaRrE96/Tucil3_13522130