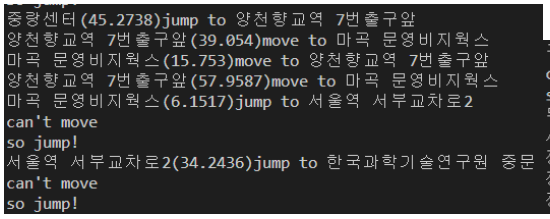
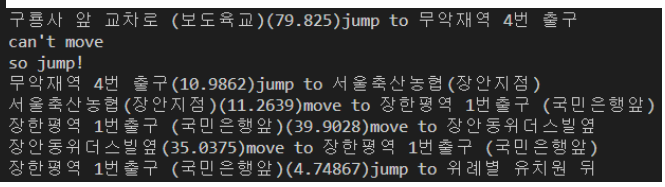
<Data Structure 2021 Midterm Project: PageRank on Graph 구현 및 실험 보고서>

숭실대학교 AI융합학부 20201786 김성연

1. Random Walker (Task1, Task4)



<그림 1 가중치가 없는 그래프 실행>



<그림 2 가중치가 있는 그래프 실행>

Random Walker가 10%의 확률로 jump를 하도록 설정하여 실행하였다. 위 그림들을 보면 알 수 있듯이 Random Walker가 10이하의 랜덤 값을 가지게 되면 jump를 실행하고 10초과의 랜덤 값을 가지게 되면 move를 실행한다. 하지만 중간에 랜덤 값이 10초과임에도 불구하고 jump가 실행되는 결과가 나온다. 이는 Random Walker가 move를 실행할 수 없는 노드 즉, 인접 노드가 존재하지 않는 노드에 위치해 있기 때문인데 이 상황에서 두 가지의 해결법을 시도했다. 이 두 가지 방법에 대해서는 '3. 이 외의 실험'에서 소개한다. 결과적으로 인접 노드가 존재하지 않으면 바로 jump를 하도록 했고 이 때문에 위 결과와 같이 10초과의 랜덤 값에서도 jump가 실행됨을 볼 수 있다.

2. PageRank (Task2, Task3, Task5, Task6, Task7)

```
<pagerank>
R2-D2 : 0.01333 QUI-GON : 0.10000 NUTE GUNRAY : 0.04333 PK-4 : 0.00667 TC-14 : 0.03000 OBI-W
AN : 0.02667 DOFINE : 0.01667 RUNE : 0.03333 TEY HOW : 0.03000 EMPEROR : 0.02000 CAPTAIN PAN
AKA : 0.02667 SIO BIBBLE : 0.01000 JAR JAR : 0.05667 TARPALS : 0.00333 BOSS NASS : 0.01667 P
ADME : 0.04333 RIC OLIE : 0.01667 WATTO : 0.01667 ANAKIN : 0.07667 SEBULBA : 0.02667 JIRA :
0.00667 SHMI : 0.03667 C-3PO : 0.02333 DARTH MAUL : 0.00667 KITSTER : 0.03667 WALD : 0.02000
 FODE/BEED : 0.00667 JABBA : 0.03333 GREEDO : 0.02000 VALORUM : 0.01667 MACE WINDU : 0.02000
 KI-ADI-MUNDI : 0.04000 YODA : 0.05000 RABE : 0.03333 BAIL ORGANA : 0.01333 GENERAL CEEL : 0
.00333 BRAVO TWO : 0.00667 BRAVO THREE : 0.01333
모든 노드의 pagerank 합 : 1.00000
가장 중요 vertex : QUI-GON (0.10000)
```

<그림 3 가중치와 방향성이 없는 그래프 실행 (walking length=300)>

```
<pagerank>
R2-D2 : 0.04667 QUI-GON : 0.16333 NUTE GUNRAY : 0.02333 PK-4 : 0.00333 TC-14 : 0.00667 OBI-W
AN : 0.05667 DOFINE : 0.00000 RUNE : 0.01333 TEY HOW : 0.00333 EMPEROR : 0.03333 CAPTAIN PAN
AKA : 0.04000 SIO BIBBLE : 0.00000 JAR JAR : 0.09667 TARPALS : 0.01000 BOSS NASS : 0.03667 P
ADME : 0.08000 RIC OLIE : 0.00000 WATTO : 0.02000 ANAKIN : 0.13000 SEBULBA : 0.00333 JIRA :
0.02000 SHMI : 0.02667 C-3PO : 0.00667 DARTH MAUL : 0.01333 KITSTER : 0.01667 WALD : 0.01333
 FODE/BEED : 0.00667 JABBA : 0.01000 GREEDO : 0.00667 VALORUM : 0.01333 MACE WINDU : 0.02667
 KI-ADI-MUNDI : 0.02000 YODA : 0.02333 RABE : 0.01333 BAIL ORGANA : 0.01333 GENERAL CEEL : 0
.00333 BRAVO TWO : 0.00000 BRAVO THREE : 0.00000
모든 노드의 pagerank 합 : 1.00000
가장 중요 vertex : QUI-GON (0.16333)
```

<그림 4 가중치가 있고 방향성이 없는 그래프 실행 (walking length=300)>

```
<pagerank>
R2-D2 : 0.02200 QUI-GON : 0.08940 NUTE GUNRAY : 0.04220 PK-4 : 0.00710 TC-14 : 0.02360 OBI-WA
N : 0.04540 DOFINE : 0.01450 RUNE : 0.01550 TEY HOW : 0.01450 EMPEROR : 0.04450 CAPTAIN PANAK
A : 0.03320 SIO BIBBLE : 0.02710 JAR JAR : 0.06910 TARPALS : 0.00840 BOSS NASS : 0.02070 PADM
E : 0.06040 RIC OLIE : 0.02460 WATTO : 0.02170 ANAKIN : 0.07550 SEBULBA : 0.02100 JIRA : 0.01
020 SHMI : 0.02840 C-3PO : 0.02140 DARTH MAUL : 0.01720 KITSTER : 0.02980 WALD : 0.01780 FODE
/BEED : 0.01150 JABBA : 0.02240 GREEDO : 0.01120 VALORUM : 0.01710 MACE WINDU : 0.02180 KI-AD
I-MUNDI : 0.02170 YODA : 0.02280 RABE : 0.01870 BAIL ORGANA : 0.00910 GENERAL CEEL : 0.01660
BRAVO TWO : 0.01120 BRAVO THREE : 0.01070
모든 노드의 pagerank 합 : 1.00000
가장 중요 vertex : QUI-GON (0.08940)
```

<그림 5 가중치와 방향성이 없는 그래프 실행 (walking length=10000)>

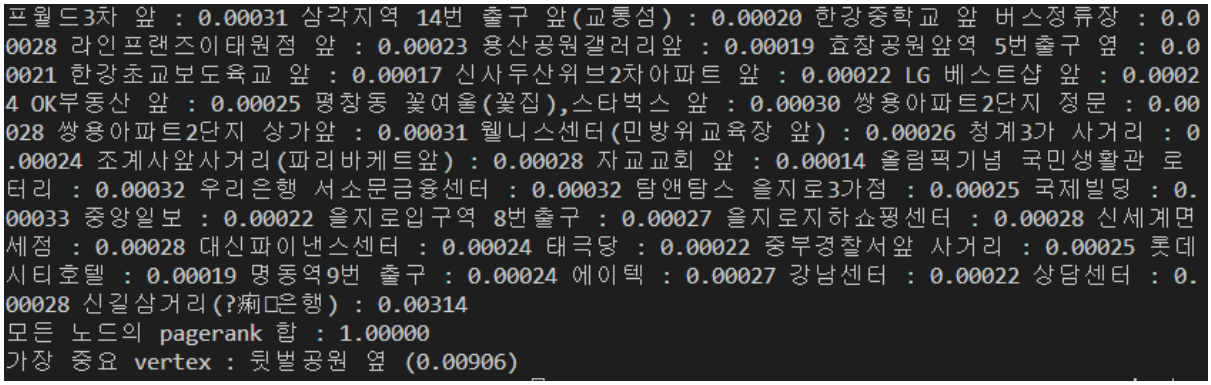
```
<pagerank>
R2-D2 : 0.04380 QUI-GON : 0.14800 NUTE GUNRAY : 0.04220 PK-4 : 0.00420 TC-14 : 0.01250 OBI-WA
N : 0.06570 DOFINE : 0.00900 RUNE : 0.02660 TEY HOW : 0.00930 EMPEROR : 0.03820 CAPTAIN PANAK
A : 0.04070 SIO BIBBLE : 0.01370 JAR JAR : 0.08220 TARPALS : 0.00440 BOSS NASS : 0.01270 PADM
E : 0.07610 RIC OLIE : 0.02030 WATTO : 0.01980 ANAKIN : 0.10550 SEBULBA : 0.01020 JIRA : 0.00
640 SHMI : 0.02860 C-3PO : 0.01180 DARTH MAUL : 0.01580 KITSTER : 0.01270 WALD : 0.01050 FODE
/BEED : 0.00570 JABBA : 0.00840 GREEDO : 0.00530 VALORUM : 0.00960 MACE WINDU : 0.01940 KI-AD
I-MUNDI : 0.01740 YODA : 0.02110 RABE : 0.00830 BAIL ORGANA : 0.00680 GENERAL CEEL : 0.00690
BRAVO TWO : 0.01220 BRAVO THREE : 0.00800
모든 노드의 pagerank 합 : 1.00000
가장 중요 vertex : QUI-GON (0.14800)
```

<그림 6 가중치가 있고 방향성이 없는 그래프 실행 (walking length=10000)>

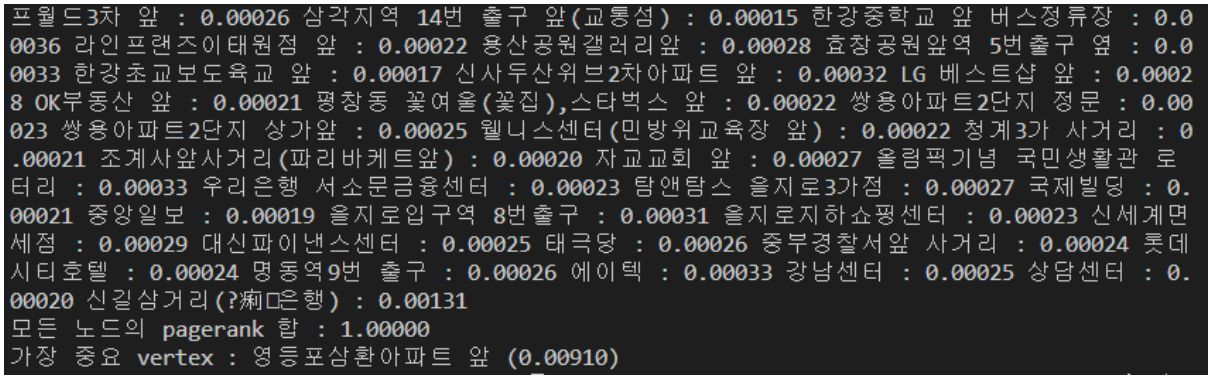
walking length 값을 조절하여 어느 정도의 수치가 되면 Random Walker가 전체 노드를 골고루 지날 수 있는지 확인한 결과, 위 실험에서 사용한 "starwars-episode-1-interactions-allCharacters-nodes.tvs"와 "starwars-episode-1-interactions-allCharacters-links.tvs"의 경우 가중치가 없을 때, length가 300정도의 값을, 가중치가 있을 때, 500정도의 값을 가져야 전체 노드를 거칠 수 있었다. 그리고 같은 데이터로 length 값이 어느 정도로 커져야 노드 간의 PageRank 차이가 최소가 되는지 확인한 결과, 가중치가 없을 때, length의 값이 10000 정도의 수치를 가져야 각 PageRank가 최소 0.008에서 최대 0.08, 가중치가 있을 때, 최소 0.005에서 최대 0.14까지가 되어 그 차이가 최소가 됨을 확인할 수 있었다. 그 이상으로 length의 값을 설정하더라도 10000에서의 PageRank 차이와 비슷함을 확인했다. (위 그림3, 그림4의 경우 length의 값을 300으로, 그림5, 그림6의 경우 10000으로 설정하였다.)

그리고 위 실험 결과를 보면 알 수 있듯이 가중치가 없을 때와 있을 때, 각 노드의 PageRank간의 차이가 드러난다. 우선 length 값을 300으로 설정했을 때의 결과처럼 가중치가 있을 때는 length 값 300으로 모든 노드를 거쳐갈 수 있었지만 가중치가 있는 경우 거쳐갈 수 없는 노드들이 발생했다. 또 length 값을 10000으로 설정한 실험에서 볼 수 있듯이 가중치가 없을 때, 0.02의 PageRank를 갖던 노드가 가중치가 생기니 더 작아진 0.008의 PageRank를 가졌고, 가중치가 없을 때 0.08의 PageRank를 갖던 노드가 가중치가 생기니 더 커진 0.14의 PageRank를 가졌다.

가중치가 없는 경우와 있는 경우의 이러한 차이들을 통해서 가중치가 Random Walker의 이동 범위와 각 노드의 중요도에 큰 영향을 미침을 확인할 수 있었다. 이는 방향성이 있는 그래프에서도 비슷한 결과를 보였는데,



<그림 7 가중치가 없고 방향성이 있는 그래프 실행 (walking length=100000)>



<그림 8 가중치와 방향성이 있는 그래프 실행 (walking length=100000)>

그림7과 그림8을 보면 알 수 있듯이 가중치가 없을 때와 있을 때의 중요 노드가 서로 다르다. 그리고 마지막 노드인 신길삼거리를 보면 가중치가 없을 때와 있을 때의 PageRank값이 3배나 차이를 보였다.

3. 이 외의 실험

1) 확률에만 의존하여 jump 혹은 move를 한다면 Random Walker 수행 시간 결과가 어떻게 다를까? (고립된 노드에서 점프의 효율성 파악)

방향성이 없는 그래프는 방향성이 있는 그래프에 비해 고립될 확률이 현저히 낮기 때문에 방향성이 있는 그래프로 실험하였고(가중치 없음), 실험을 하기 위해 Random Walker의 코드를 수정하였다.

(실험을 위해 수정한 Random Walker의 코드는 movejump_test.cpp라는 파일로 첨부하였다.)

고립된 노드에서 점프의 시간적 효율성을 보고자 했으므로 walking length의 값은 move 또는 jump가 수행될 경우만 줄어들도록 했다. 그리고 두 경우 모두 walking length=10000, jump 확률=0.1(10%)로 실험을 진행했다. 실험 결과는 다음 그림9, 그림10과 같다. 확률에만 의존하여 jump를 할 수 있는 경우, 그림9의 결과처럼 move를 실행할 수 없으면 jump의 확률이 나올 때까지 다시 확률을 받기 때문에 반복문이 돌아가는 횟수가 일정하지 않고 무수히 커질 수 있다. 하지만 그림10처럼 move를 실행할 수 없을 때 바로 jump를 실행하는 경우, jump를 기다리며 확률을 계

속 다시 받아올 일이 없기 때문에 반복문이 돌아가는 횟수가 walking length와 일치하게 된다. 이렇게 반복문을 실행하는 횟수에 큰 차이를 보이기 때문인지 확률에만 의존하여 jump를 하는 Random Walker의 경우 실행하는데 32859869ms의 시간이 소요되었고, 인접 노드가 없다면 바로 jump를 실행하는 Random Walker의 경우 실행하는데 6723772ms의 시간이 소요되었다.

```
용마산역 1번출구(1.59694)jump to 내방역 8번출구 앞
can't move
can't move
can't move
can't move
can't move
can't move
can't move
can't move
can't move
can't move
can't move
내방역 8번출구 앞(0.528456)jump to 동일로58길 입구
Random Walker 소요 시간 : 32859869
```

<그림 9 확률에만 의존하여 jump 할 수 있는 경우>

```
한강 현대아파트 건너편(61.5245)jump to 서울북부수도사업소
can't move
so jump!
서울북부수도사업소(26.0928)jump to 동대문중학교 옆
can't move
so jump!
동대문중학교 옆(97.4116)jump to 신양초교앞 교차로
can't move
so jump!
신양초교앞 교차로(43.3026)jump to 아만티호텔 앞
can't move
so jump!
아만티호텔 앞(70.495)jump to 노원문화예술회관
Random Walker 소요 시간 : 6723772
```

<그림 10 인접 노드가 없다면 jump를 실행하는 경우>

두 경우의 실행 소요 시간은 약5배가 차이가 나며, 이는 walking length가 커질수록 더 큰 차이를 낸다. 따라서 확률에만 의존하여 jump를 하는 것보다는 move의 실행 가능 여부를 판단한 후 jump 여부를 판단하는 것이 시간적 효율성이 훨씬 뛰어나다는 것을 볼 수 있다.

2) Random Walker가 이동한 경로를 바탕으로 찾아낸 가장 중요한 노드에서 다시 Random Walker를 시작하는 방식으로 모든 노드가 Random Walker의 시작 노드가 될 때까지 실행해보기

```
시작 vertex : R2-D2
가장 중요 vertex : WATTO (0.08000)
가장 중요 vertex : QUI-GON (0.08333)
가장 중요 vertex : TEY HOW (0.07333)
가장 중요 vertex : SHMI (0.05333)
가장 중요 vertex : TC-14 (0.05000)
가장 중요 vertex : RIC OLIE (0.05000)
가장 중요 vertex : RUNE (0.04333)
가장 중요 vertex : YODA (0.06000)
가장 중요 vertex : WALD (0.04000)
가장 중요 vertex : OBI-WAN (0.04000)
가장 중요 vertex : PADME (0.05333)
가장 중요 vertex : NUTE GUNRAY (0.04333)
가장 중요 vertex : EMPEROR (0.03667)
가장 중요 vertex : SIO BIBBLE (0.03000)
가장 중요 vertex : JAR JAR (0.04000)
가장 중요 vertex : JABBA (0.03333)
가장 중요 vertex : RABE (0.02333)
가장 중요 vertex : SEBULBA (0.03667)
가장 중요 vertex : VALORUM (0.03333)
가장 중요 vertex : MACE WINDU (0.02333)
가장 중요 vertex : BRAVO TWO (0.02333)
가장 중요 vertex : KITSTER (0.03000)
가장 중요 vertex : FODE/BEED (0.02667)
가장 중요 vertex : GENERAL CEEL (0.02667)
가장 중요 vertex : PK-4 (0.02000)
가장 중요 vertex : JIRA (0.02333)
가장 중요 vertex : TARPALS (0.01667)
가장 중요 vertex : BRAVO THREE (0.02667)
가장 중요 vertex : CAPTAIN PANAKA (0.01667)
가장 중요 vertex : BOSS NASS (0.02000)
가장 중요 vertex : DARTH MAUL (0.02000)
가장 중요 vertex : DOFINE (0.02000)
가장 중요 vertex : GREEDO (0.01667)
가장 중요 vertex : C-3PO (0.01667)
가장 중요 vertex : ANAKIN (0.01667)
가장 중요 vertex : BAIL ORGANA (0.01667)
가장 중요 vertex : KI-ADI-MUNDI (0.02000)
PS C:\Users\user\Desktop\Data Structure>
```

실험을 위해 기존 PageRank 함수의 가장 중요 노드 찾는 코드를 수정하였고, 수정한 함수를 이용해 All_node_rank 라는 새로운 함수를 만들었다. (실험에 사용한 코드는 All_node_test.cpp라는 파일로 첨부하였다.)

입력받은 노드에서 시작한 Random Walker의 가장 중요한 노드를 찾은 후 찾은 노드를 시작으로 다시 Random Walker를 진행하게 하는 것을 반복하였는데, 이 때 이미 시작 노드의 경험이 있는 노드는 가장 중요한 노드 즉, 다시 시작 노드가 될 수 없게 설정하였다. 그 이유는 맨 처음에 주어진 시작 노드의 입장에서 모든 노드들을 한 번씩 지나야 할 때, 이동할 순서를 효율적으로 정하고 싶었기 때문이다.

이런 목적을 가지고 실험을 진행한 결과 그림11과 같이 모든 노드가 한 번 씩 출력되는 것을 확인할 수 있었고, 그것이 Random Walker가 시작 노드에서 모든 노드를 한

<그림 11 'R2-D2' 노드에서 시작한 All_node_rank 실행결과>

번 씩 지날 때, 각 위치에서 가장 효율적인 선택을 하는 것이라고 판단했다.

이런 실험을 조금 더 실생활에 적용하면 여행지를 입력했을 때, 여행 일정을 효율적으로 세워주는 프로그램을 만들 수 있지 않을까 생각이 들었다.

4. 느낀 점

처음에는 무엇부터 시작해야 할 지 너무 막막하기만 했었다. 일단 그래프부터 그리자는 생각으로 시작을 하니 무언가 하나씩 해결되면서 프로젝트의 재미를 느낄 수 있었던 것 같다. 사실 Task를 위한 코드만 다 구현했을 때도 무척 뿌듯했는데, 내가 작성한 코드를 이용해 여러 실험을 해보니까 그 뿌듯함이 배가 되었던 것 같다. 또 이것을 어디에 적용해 볼 수 있을까 고민하는 시간이 즐거웠다. 처음에는 대단한 실험을 해보겠다는 포부로 4시간 동안 여러 데이터 자료들을 찾아보았지만 내가 작성한 코드(데이터 저장 코드)에 맞는 데이터 자료들을 찾기가 너무 어려워서 주어진 코드로 여러 실험을 해보는 방향으로 계획을 수정했었다. 그래서인지 아직 다양한 데이터를 사용해보지 못한 아쉬움이 남은 것 같다. 프로젝트는 이렇게 끝이 났지만 이후에도 이 코드를 수정해서 다양한 형태의 데이터를 저장할 수 있게 해보고 싶다.