Data Structure Project

Project #1

담당교수 : 심 동규 교수님

제출일자 : 2019.10.09

학　　과 : 컴퓨터정보공학부

학　　번 : 2014722038

이　　름 : 이 정 호

1. Introduction

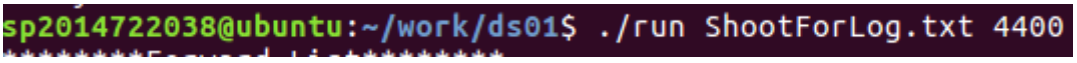
이번 프로젝트는 BST(Binary Search Tree)를 이용하여 축구 구단 관리 프로그램을 구현하는 프로젝트입니다. 감독의 입장으로서 선수 명단(ShootForLog.txt)으로 구단주에게 받은 돈으로 가장 강력한 팀을 꾸릴 수 있도록 한다. 선수 명단에 저장된 선수에 대한 정보는 각 선수의 이름(name), 포지션(position), 이적료(fee), 능력치(ability)로 구성 되어있다. 이름은 영문으로 저장되어 있으며 포지션의 종류는 forward(공격수), Midfielder(미드필더), Defender(수비수), Goalkeeper(골키퍼)로 총 4개의 포지션이 있고 이적료는 정수의 값으로 저장을 받습니다. 또한 같은 포지션 내에 이적료가 같은 선수는 존재하지 않으며 이적료와 능력치는 비례하지 않는다. 능력치의 범위는 1~99이며 같은 포지션내에서 능력치가 서로 같은 선수는 존재 하지 않는다.

본프로젝트의 핵심 요소는 BST의 기능을 구현하며 Best_team을 만드는 것에 있다고 생각한다. Bestteam은 4명의 선수로 구성 되어있고 공격수 1명, 미드필더 1명, 수비수 1명, 골키퍼 1명으로 각 포지션별로 한 명을 뽑는다. 뽑은 선수들의 이적료 합은 구단주가 준 돈(m_budget)보다 작거나 같아야 하며 능력치의 합이 가장 높은 team이 bestteam이 된다. 만약 능력치의 합이 가장 높은 두 개 이상의 팀이 존재 한다면 이적료의 합이 가장 적은 팀이 best_team이 된다.

2. Algorithm

1) Setup

프로그램 실행 시 두개의 입력이 존재하는데 처음은 ShootForLog.txt로서 선수의 명단을 불러오고 두 번째는 구단주에게 받은 돈의 액수(억)이다. 여기서 두 번째 입력 즉 구단주에게 받은 돈은 최소 1664억 이상이다.



<첫 번째 입력과 두 번째 입력을 넣은 모습>

프로그램 실행 시 선수 명단을 읽고 선수들의 정보를 포지션에 따라 4개의 BST에 저장하도록 한다. 각 BST의 루트는 선수 명단에 데이터 순으로 처음 나타나는 포지션에 해당하는 선수가 해당 포지션 별 BST의 루트 노드가 된다.

```
if (str_position == " Forward") //if fw
{
		fwBST.insert(*player_data); //insert
		fw_arr[fw_c].insert_arr(*player_data);
		fw_c++;
}
else if (str_position == " Goalkeeper") //if gk
{
		gkBST.insert(*player_data); //insert
		gk_arr[gk_c].insert_arr(*player_data);
		gk_c++;
}
else if (str_position == " Midfielder")//if mf
{
		mfBST.insert(*player_data); //insert
		mf_arr[mf_c].insert_arr(*player_data);
		mf_c++;
}
else if (str_position == " Defender") //if df
{
		dfBST.insert(*player_data); //insert
		df_arr[df_c].insert_arr(*player_data);
		df_c++;
```

먼저 텍스트 파일을 읽어서 각 포지션별로 insert를 해준다.

```
TreeNode * p = m_root;
TreeNode * pp = NULL;
if (p == NULL)
m_root = player;
return;
```

움직여야할 노드를 만들어주고 그 노드의 부모노드까지 같이 만들어 준다.만약 루트가 NULL이면 즉 데이터가 없으면 루트에 플레이어를 넣고 리턴한다.

```
else |
while (p)
{
pp = p;
if (data.m_ability < p->m_data.m_ability)
p = p->m_left;
else if (data.m_ability > p->m_data.m_ability)
p = p->m_right;
}
if (data.m_ability < pp->m_data.m_ability)
pp->m_left = player;
else
pp->m_right = player;
}
```

루트 노드가 비어있지 않으면 들어가야할 노드의 위치를 탐색하는 while문을 만들어 준다. 만약 insert 하는 데이터의 값이 현재 p가 있는 위치보다 작으면 왼쪽으로 가고 크면 오른쪽으로 가게 하는 트리의 습성을 이용한다. 위치를 찾고 리프노드를 찾으면 리프노드보다 작으면 왼쪽에 데이터를 넣어주고 크면 오른쪽에 데이터를 넣어준다.

2) print Players

데이터를 저장 후 데이터를 inorder방식으로 순회하며 선수 목록을 출력한다.. 출력 순서는 fw, mf, df, gk이다.

```
//inorder
while (tree.m_root != NULL)//while root is NULL
{
        os << tree.m_root;
        return os;
}
```

맨 처음 operation 부분. 먼저 루트가 널일때까지 루트를 넣어서 재귀적으로 도는 함수다.

```
if (node != NULL)
{
        os << node->m_left;
        os << node->m_data << std::endl;
        os << node->m_right;
}
return os;
```

노드가 맨 왼쪽으로 이동을 한 뒤 그 다음 출력부분을 실행 해주고 왼쪽 노드가 없으면 오른쪽 노드로 가는 방식이다.

```
os << "(node.m_name: " << node.m_name << "), " //print
        << "(node.m_position: " << node.m_position << "), "
        << "(node.m_transfer_fee: " << node.m_transfer_fee << "), "
        << "(node.m_ability: " << node.m_ability << ")";
```

출력 부분입니다. Visit이라고도 할 수 있습니다.

3) Search the Best Team & Delete best team player

각 포지션 별로 선수 한 명을 선별하며 현재 보유한 재산으로 구매 가능한 가장 강력한 팀을 찾아낸다. 탐색이 끝나면 포지션 별로 가장 강력한 팀의 존재하는 선수들의 정보를 제거한다. 이 후 가장 강력한 팀을 출력하고 포지션 별로 선수 목록을 출력한다. 여기서 선수 목록은 정보가 제거 된 후이다.

BST로부터 선수를 제거할 때 제거될 노드가 left 서브트리와 right 서브트리를 같는 경우 left서브트리 내에서 가장 큰 능력치를 갖는 노드를 찾고 해당 노드를 제거될 노드에 위치시키도록 한다.

베스트 팀을 뽑는 코드는 다음과 같다.

```
for (int z = 0; z < fw_c; z++)
	{
		for (int x = 0; x < mf_c; x++) //for all mf's player
		{
			for (int y = 0; y < df_c; y++) //for all df's player
			{
				for (int w = 0; w < gk_c; w++) //for all gk's player
				{
```

각 포지션별로 한 명씩 돌아가면서 만들 수 있는 모든 조합을 만들면서 비교하는 알고리즘이다.

```
if (fw_arr[z].m_root->m_data.m_transfer_fee
		+ mf_arr[x].m_root->m_data.m_transfer_fee
		+ df_arr[y].m_root->m_data.m_transfer_fee
		+ gk_arr[w].m_root->m_data.m_transfer_fee <= m_budget)
```

첫 번째 if 문에는 각배열을 돌면서 선수의 이적료 합이 구단주에게 받은 돈(m_budget)보다 작아야 다음 조건으로 갈 수 있다.

```
if (best_team.sum_ability <=
		fw_arr[z].m_root->m_data.m_ability
		+ mf_arr[x].m_root->m_data.m_ability
		+ df_arr[y].m_root->m_data.m_ability
		+ gk_arr[w].m_root->m_data.m_ability)
```

두 번째 if 문에는 능력치의 비교이다 첫 번째 조건을 만족한 팀의 능력치의 합이 기존에 있던 sum_ability보다 크거나 같으면 다음 if으로 넘어간다.

```
if (best_team.sum_ability ==
		fw_arr[z].m_root->m_data.m_ability
		+ mf_arr[x].m_root->m_data.m_ability
		+ df_arr[y].m_root->m_data.m_ability
		+ gk_arr[w].m_root->m_data.m_ability)
```

만약 능력치가 같을 경우일 때를 먼저 고려해줬다.

```
if (fw_arr[z].m_root->m_data.m_transfer_fee
		+ mf_arr[x].m_root->m_data.m_transfer_fee
		+ df_arr[y].m_root->m_data.m_transfer_fee
		+ gk_arr[w].m_root->m_data.m_transfer_fee
		< best_team.fw.m_transfer_fee + best_team.df.m_transfer_fee
		+ best_team.mf.m_transfer_fee + best_team.gk.m_transfer_fee)
```

만약 능력치가 같더라도 더 낮은 이적료를 가진 팀이 더욱 강력한 팀이기에 비교를 해주고 다음 조건으로 넘어간다.

```
best_team.fw = fw_arr[z].m_root->m_data;
best_team.mf = mf_arr[x].m_root->m_data;
best_team.df = df_arr[y].m_root->m_data;
best_team.gk = gk_arr[w].m_root->m_data;
```

능력치가 같고 더 낮은 이적료를 가진 팀이 있다면 best팀의 선수들을 현재의 선수들로 바꿔준다. 만약 두 번째 능력치가 같지 않고 더 크다면 같을 때를 고려하지 않고 바로 best팀의 선수들을 현재의 선수들로 바꿔준다.

3. Result Screen

```
sp2014722038@ubuntu:~/work/ds01$ make
g++ -std=c++11 -g -o run DS_Project1.cpp SoccerPlayerData.cpp
sp2014722038@ubuntu:~/work/ds01$ ./run ShootForLog.txt 4400
```

Make로 실행파일(run)을 만들어 준 뒤 첫번째 인자 즉 argv[1]에는 선수 명단인 ShootForLog.txt를 넣어주고 두 번째 인자(argv[2])에는 감독에게 받은 이적료를 넣어 줬다.

```
sp2014722038@ubuntu:~/work/ds01$ ./run ShootForLog.txt 4400
********Forward List********
(node.m_name: Ronaldo), (node.m_position:  Forward), (node.m_transfer_fee: 250), (node.m_ability: 63)
(node.m_name: Sterling), (node.m_position:  Forward), (node.m_transfer_fee: 675), (node.m_ability: 85)
(node.m_name: Mbappe), (node.m_position:  Forward), (node.m_transfer_fee: 2475), (node.m_ability: 87)
(node.m_name: Griezmann), (node.m_position:  Forward), (node.m_transfer_fee: 809), (node.m_ability: 88)
(node.m_name: Kane), (node.m_position:  Forward), (node.m_transfer_fee: 844), (node.m_ability: 89)
(node.m_name: Neymar), (node.m_position:  Forward), (node.m_transfer_fee: 1119), (node.m_ability: 90)
(node.m_name: Lewandowski), (node.m_position:  Forward), (node.m_transfer_fee: 950), (node.m_ability: 91)
(node.m_name: Salah), (node.m_position:  Forward), (node.m_transfer_fee: 960), (node.m_ability: 92)
(node.m_name: Hazard), (node.m_position:  Forward), (node.m_transfer_fee: 677), (node.m_ability: 93)
(node.m_name: Dybala), (node.m_position:  Forward), (node.m_transfer_fee: 870), (node.m_ability: 94)
(node.m_name: Suarez), (node.m_position:  Forward), (node.m_transfer_fee: 1250), (node.m_ability: 95)
(node.m_name: Seung-woo), (node.m_position:  Forward), (node.m_transfer_fee: 800), (node.m_ability: 96)
(node.m_name: Jisung), (node.m_position:  Forward), (node.m_transfer_fee: 1000), (node.m_ability: 97)
(node.m_name: Son), (node.m_position:  Forward), (node.m_transfer_fee: 1200), (node.m_ability: 98)
(node.m_name: Messi), (node.m_position:  Forward), (node.m_transfer_fee: 1300), (node.m_ability: 99)
**************************
********Midflder List********
(node.m_name: Vidal), (node.m_position:  Midfielder), (node.m_transfer_fee: 389), (node.m_ability: 75)
(node.m_name: Robben), (node.m_position:  Midfielder), (node.m_transfer_fee: 589), (node.m_ability: 85)
(node.m_name: Fabregas), (node.m_position:  Midfielder), (node.m_transfer_fee: 721), (node.m_ability: 86)
(node.m_name: Toure), (node.m_position:  Midfielder), (node.m_transfer_fee: 1011), (node.m_ability: 87)
(node.m_name: Eriksen), (node.m_position:  Midfielder), (node.m_transfer_fee: 947), (node.m_ability: 88)
(node.m_name: Schweinsteiger), (node.m_position:  Midfielder), (node.m_transfer_fee: 713), (node.m_ability: 91)
(node.m_name: Silva), (node.m_position:  Midfielder), (node.m_transfer_fee: 342), (node.m_ability: 92)
(node.m_name: Modric), (node.m_position:  Midfielder), (node.m_transfer_fee: 867), (node.m_ability: 93)
(node.m_name: Kroos), (node.m_position:  Midfielder), (node.m_transfer_fee: 762), (node.m_ability: 94)
(node.m_name: Busquets), (node.m_position:  Midfielder), (node.m_transfer_fee: 321), (node.m_ability: 95)
(node.m_name: Zidane), (node.m_position:  Midfielder), (node.m_transfer_fee: 1115), (node.m_ability: 96)
(node.m_name: Iniesta), (node.m_position:  Midfielder), (node.m_transfer_fee: 845), (node.m_ability: 97)
(node.m_name: Pogba), (node.m_position:  Midfielder), (node.m_transfer_fee: 2581), (node.m_ability: 98)
(node.m_name: Kang in), (node.m_position:  Midfielder), (node.m_transfer_fee: 900), (node.m_ability: 99)
**************************
```

```
********Defender List********
(node.m_name: Luiz), (node.m_position:  Defender), (node.m_transfer_fee: 741), (node.m_ability: 65)
(node.m_name: Maicon), (node.m_position:  Defender), (node.m_transfer_fee: 777), (node.m_ability: 77)
(node.m_name: Neville), (node.m_position:  Defender), (node.m_transfer_fee: 1212), (node.m_ability: 80)
(node.m_name: Pique), (node.m_position:  Defender), (node.m_transfer_fee: 813), (node.m_ability: 81)
(node.m_name: Zanetti), (node.m_position:  Defender), (node.m_transfer_fee: 819), (node.m_ability: 85)
(node.m_name: Lahm), (node.m_position:  Defender), (node.m_transfer_fee: 817), (node.m_ability: 88)
(node.m_name: Carlos), (node.m_position:  Defender), (node.m_transfer_fee: 999), (node.m_ability: 89)
(node.m_name: Vidic), (node.m_position:  Defender), (node.m_transfer_fee: 349), (node.m_ability: 92)
(node.m_name: Ramos), (node.m_position:  Defender), (node.m_transfer_fee: 913), (node.m_ability: 93)
(node.m_name: Maldini), (node.m_position:  Defender), (node.m_transfer_fee: 498), (node.m_ability: 94)
(node.m_name: Yeonggwon), (node.m_position:  Defender), (node.m_transfer_fee: 1218), (node.m_ability: 95)
(node.m_name: Nesta), (node.m_position:  Defender), (node.m_transfer_fee: 1050), (node.m_ability: 96)
(node.m_name: Puyol), (node.m_position:  Defender), (node.m_transfer_fee: 924), (node.m_ability: 97)
(node.m_name: Alves), (node.m_position:  Defender), (node.m_transfer_fee: 247), (node.m_ability: 98)
(node.m_name: van Dijk), (node.m_position:  Defender), (node.m_transfer_fee: 1450), (node.m_ability: 99)
*****************************
********Goalkeeper List********
(node.m_name: Jeong Sung-Ryong), (node.m_position:  Goalkeeper), (node.m_transfer_fee: 846), (node.m_ability: 54)
*******************************
```

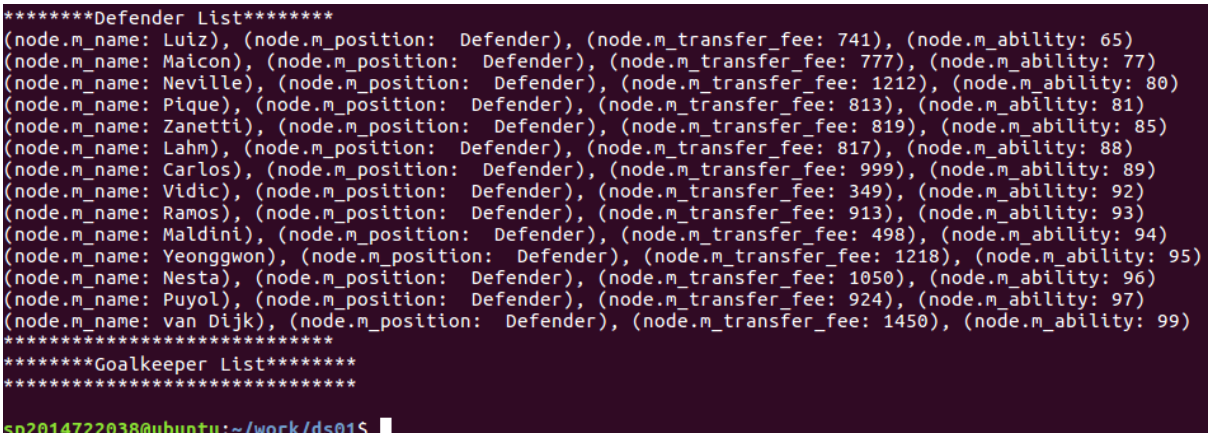
맨 처음 출력은 공격수, 미드필더, 수비수, 골키퍼 순으로 inorder순회로 출력이 된 모습이다. 능력치 기준으로 노드가 생성이 됐기 때문에 작은 능력치부터 큰 능력치로 오름차순 출력이 된 것을 확인할 수 있다.

```
Best Players
(node.m_name: Messi), (node.m_position:  Forward), (node.m_transfer_fee: 1300), (node.m_ability: 99)
(node.m_name: Kang in), (node.m_position:  Midfielder), (node.m_transfer_fee: 900), (node.m_ability: 99)
(node.m_name: Alves), (node.m_position:  Defender), (node.m_transfer_fee: 247), (node.m_ability: 98)
(node.m_name: Jeong Sung-Ryong), (node.m_position:  Goalkeeper), (node.m_transfer_fee: 846), (node.m_ability: 54)
sum_transfer_fee 3293
sum_ability 350

-----
The Transfer window close
```

다음으로 출력 된 것은 best team의 선수 명단이다. 역시 공격수, 미드필더, 수비수, 골키퍼 순으로 출력이 되었고 4400의 인자를 주었을 때 프로젝트 제안서와 같은 결과를 얻은 것을 확인할 수 있다.

```
********Forward List********
(node.m_name: Ronaldo), (node.m_position:  Forward), (node.m_transfer_fee: 250), (node.m_ability: 63)
(node.m_name: Sterling), (node.m_position:  Forward), (node.m_transfer_fee: 675), (node.m_ability: 85)
(node.m_name: Mbappe), (node.m_position:  Forward), (node.m_transfer_fee: 2475), (node.m_ability: 87)
(node.m_name: Griezmann), (node.m_position:  Forward), (node.m_transfer_fee: 809), (node.m_ability: 88)
(node.m_name: Kane), (node.m_position:  Forward), (node.m_transfer_fee: 844), (node.m_ability: 89)
(node.m_name: Neymar), (node.m_position:  Forward), (node.m_transfer_fee: 1119), (node.m_ability: 90)
(node.m_name: Lewandowski), (node.m_position:  Forward), (node.m_transfer_fee: 950), (node.m_ability: 91)
(node.m_name: Salah), (node.m_position:  Forward), (node.m_transfer_fee: 960), (node.m_ability: 92)
(node.m_name: Hazard), (node.m_position:  Forward), (node.m_transfer_fee: 677), (node.m_ability: 93)
(node.m_name: Dybala), (node.m_position:  Forward), (node.m_transfer_fee: 870), (node.m_ability: 94)
(node.m_name: Suarez), (node.m_position:  Forward), (node.m_transfer_fee: 1250), (node.m_ability: 95)
(node.m_name: Seung-woo), (node.m_position:  Forward), (node.m_transfer_fee: 800), (node.m_ability: 96)
(node.m_name: Jisung), (node.m_position:  Forward), (node.m_transfer_fee: 1000), (node.m_ability: 97)
(node.m_name: Son), (node.m_position:  Forward), (node.m_transfer_fee: 1200), (node.m_ability: 98)
*****************************
********Midflder List********
(node.m_name: Vidal), (node.m_position:  Midfielder), (node.m_transfer_fee: 389), (node.m_ability: 75)
(node.m_name: Robben), (node.m_position:  Midfielder), (node.m_transfer_fee: 589), (node.m_ability: 85)
(node.m_name: Fabregas), (node.m_position:  Midfielder), (node.m_transfer_fee: 721), (node.m_ability: 86)
(node.m_name: Toure), (node.m_position:  Midfielder), (node.m_transfer_fee: 1011), (node.m_ability: 87)
(node.m_name: Eriksen), (node.m_position:  Midfielder), (node.m_transfer_fee: 947), (node.m_ability: 88)
(node.m_name: Schweinsteiger), (node.m_position:  Midfielder), (node.m_transfer_fee: 713), (node.m_ability: 91)
(node.m_name: Silva), (node.m_position:  Midfielder), (node.m_transfer_fee: 342), (node.m_ability: 92)
(node.m_name: Modric), (node.m_position:  Midfielder), (node.m_transfer_fee: 867), (node.m_ability: 93)
(node.m_name: Kroos), (node.m_position:  Midfielder), (node.m_transfer_fee: 762), (node.m_ability: 94)
(node.m_name: Busquets), (node.m_position:  Midfielder), (node.m_transfer_fee: 321), (node.m_ability: 95)
(node.m_name: Zidane), (node.m_position:  Midfielder), (node.m_transfer_fee: 1115), (node.m_ability: 96)
(node.m_name: Iniesta), (node.m_position:  Midfielder), (node.m_transfer_fee: 845), (node.m_ability: 97)
(node.m_name: Pogba), (node.m_position:  Midfielder), (node.m_transfer_fee: 2581), (node.m_ability: 98)
*****************************
```



마지막 출력 부분은 베스트팀에 있는 선수들이 모두 삭제가 된 후의 선수 명단이다. 공격수에서 Messi의 정보가 빠지고 미드 필더에서 Kang in의 정보가 수비수에선 Alves의 정보가 빠진 것을 확인할 수 있다. 골키퍼에는 단 한 명의 골키퍼 밖에 존재하지 않았기 때문에 Jeong Sung-Ryong이 빠진 후 다른 정보가 없는 것을 확인할 수 있다.

4. Consideration

이번 프로젝트를 진행하면서 BST를 구현하는 것에는 크게 어려움이 없었으나, operation을 이용하는 것은 처음이라 애를 먹은 기억이 있다. 하지만 이것저것 하다 보니 얼떨결에 됐고 그것을 기반으로 수행할 수 있었다. 그리고 operation에 대한 개념도 찾아보면서 공부가 되었다. 또 한가지 애를 먹었던 점은 best_team을 만드는 것 이였다. 처음엔 가성비를 따지는 건지 너무 어렵게 생각했던 나머지 생각이 잘 나지 않았고 제안서를 몇 번이나 다시 읽은 결과 그저 조건만 만족하면 되겠구나 생각이 났다 이 부분이 제일 오래 걸렸다. 마지막으로 BST의 삭제부분은 아직도 헷갈리는 부분이 있다. 개념은 어렵지 않지만 신경 써줘야 하는 부분도 많고 경우의 수도 몇 가지가 있기 때문이다. 전체적으로 코드를 구현하기보다 메모리 해제나 초기화, 사소하지만 중요한 것 들에서 애를 먹었다.