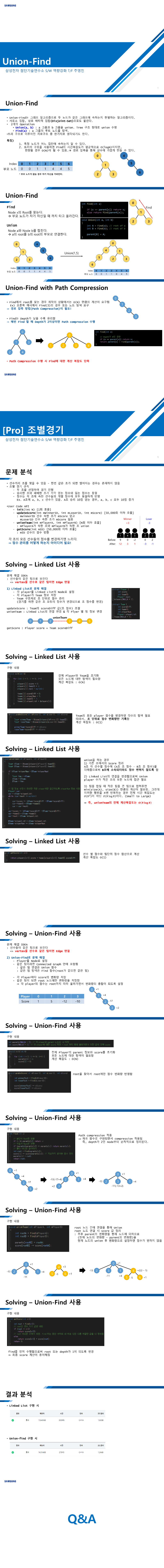
**목차   
1. 문제 풀이 강의 자료   
2. 소스 코드   
   1) Linked List  
   2) Union-Find**  
  
  
**1. 문제 풀이 강의 자료   
**  
  
**2. 소스 코드**  
1) Linked List 활용

**#define MAX\_N 100000**

**using** **namespace** std;

**struct** Player {

**int** score;

**int** teamID;

Player\* next;

}players[MAX\_N + 1];

**struct** Team {

**int** scoreDiff;

**int** teamID;

**int** playerNum;

Player\* playerList;

}teams[MAX\_N + 1];

**void** init(**int** N)

{

**for** (**int** i = 1; i <= N; i++)

{

players[i].score = 0;

players[i].teamID = i;

players[i].next = **nullptr**;

teams[i].scoreDiff = 0;

teams[i].playerNum = 1;

teams[i].teamID = i;

teams[i].playerList = &players[i];

}

}

**void** updateScore(**int** mWinnerID, **int** mLoserID, **int** mScore)

{

Team\* winnerTeam = &teams[players[mWinnerID].teamID];

Team\* loserTeam = &teams[players[mLoserID].teamID];

winnerTeam->scoreDiff += mScore;

loserTeam->scoreDiff -= mScore;

}

**void** unionTeam(**int** mPlayerA, **int** mPlayerB)

{

*// 짧은 것 뒤에 긴것 붙이기. A팀 < B팀*

Team\* ATeam = &teams[players[mPlayerA].teamID];

Team\* BTeam = &teams[players[mPlayerB].teamID];

**if** (ATeam->playerNum > BTeam->playerNum)

{

Team\* tmp = ATeam;

ATeam = BTeam;

BTeam = tmp;

}

*// A팀은 쭉 돌더라도 B팀은 전체 돌 필요 없음*

Player\* cur;

cur = ATeam->playerList;

**while** (cur->next != **nullptr**)

{

cur->score += (ATeam->scoreDiff - BTeam->scoreDiff);

cur->teamID = BTeam->teamID;

cur = cur->next;

}

cur->score += (ATeam->scoreDiff - BTeam->scoreDiff);

cur->teamID = BTeam->teamID;

cur->next = BTeam->playerList;

BTeam->playerList = ATeam->playerList;

BTeam->playerNum += ATeam->playerNum;

}

**int** getScore(**int** mID)

{

**return** players[mID].score + teams[players[mID].teamID].scoreDiff;

}

2) Union-Find 활용

**#define MAX\_N 100000**

**int** parents[MAX\_N + 1]; *// 각 player의 parent 노드의 id 저장*

**int** score[MAX\_N + 1]; *// 각 player에 대한 score 저장 - root 까지 따라 올라가면서 더한 값이 진짜 score*

*// id=x인 player의 root를 반환해주는 함수.*

**int** find(**int** x)

{

*// 본인이 Root면 반환*

**if** (x == parents[x]) **return** x;

*// 부모가 Root면 반환*

**if** (parents[parents[x]] == parents[x]) **return** parents[x];

*// 둘다 아니면 갈아타기*

**int** root = find(parents[x]);

score[x] += score[parents[x]]; *// 지금까지 쌓아둔 점수 정리*

parents[x] = root;

**return** root;

}

**void** init(**int** N)

{

**for** (**int** i = 1; i <= N; i++) {

parents[i] = i;

score[i] = 0;

}

}

**void** updateScore(**int** mWinnerID, **int** mLoserID, **int** mScore)

{

**int** winnerRoot = find(mWinnerID);

**int** loserRoot = find(mLoserID);

score[winnerRoot] += mScore;

score[loserRoot] -= mScore;

}

**void** unionTeam(**int** mPlayerA, **int** mPlayerB)

{

**int** rootA = find(mPlayerA);

**int** rootB = find(mPlayerB);

parents[rootB] = rootA;

score[rootB] -= score[rootA];

}

**int** getScore(**int** mID)

{

**int** root = find(mID);

*// root인 경우 자기 값만 리턴*

**if** (root == mID)

**return** score[mID];

*// root 아니면 더해서 리턴. find 하는 동안 어차피 내 위로 있던 다른 짜잘한 값들 다 정리됨*

**else**

**return** score[mID] + score[root];

**return** 0;

}