HW2 Dry

December 24, 2022

Into:

We built our project in 3 parts. First we created a dynamic array class thats in charge of managing all the memory in our project, this class allows us to reach a Player in O(1) on average as seen in class. The union_find class uses the dynamic array class and is exactly as we saw in class and has compression therefor the complexity is aproximate o(log*n). We use our special union_class that is specificly built for this exercise (meaning it handles players and is not template based).

dynamic array class:

dynamic_array:

Init. an empty array of linked list the init. size of this array is 10 and grows when the load factor is biger than 0.75.
O(1) aproximate

check_array:

checks that the load factor is bellow 0.75 and if it is not then we expand the array times 2 its size. O(1) aproximate.

newPlayerNode:

puts a new player in the dynamic array by allocating a new node_to_player which is the linked list that the dynamic array holds in each place in the array. We also allocate a new PlayerNode objects (the linked list in the array points to the PlayerNode object). We return back the PlayerNode (we wont lose it since the dynamic array is pointing to it).
O(1) aprox.

getPlayerNode:

We look up a specific player in dynamic array using the hash function. O(1).

hash:

Normal destribution of the elements in the array using the module function. O(1) aprox.

Union_find Class

find:

We use compression find as seen in the lectures in order to get our aprox. on average O(log*n) complexity. It is important to note that we manipulated the find funtion in order to find how many games a player has played and in order to find the get partial spirit.

unite:

This is also similare to the unite we saw in class but the only diffrence is that we must give in a NodePlayer pointer to the big team first and only then to the small team. We also merge the teams data and put them back in the big team. This is basically the same idea that we saw in class but made specific for this exercise. O(log*n) aprox. on average.

createNewPlayerSet:

We add a new player to the team if there is no root player in the team this means it is the first player we add so we make him our root player otherwise we just make him point to the root player of the team. We create the NodePlayer by calling our NewPlayerNode function which I explained in the dynamic array section. O(log*n) aprox. on average.

getSpecificPlayer:

Gets the player by calling the getPlayerNode function in the dynamic array. O(1).