Projekt

Inledning:

Denna uppgift belyser **ADT Disjunkta mängder** (som behandlas i kapitel 8 av Data Structures and Algorithm Analysis in C++ av Mark A. Weiss.)

Du ska skriva ett program som visar effekten med och utan stigkomprimering (path compression) vid operationen **find** och två olika strategier för **union**. Du ska därefter skriva en kort rapport och i denna beskriva 2 olika användningsområden för disjunkta mängder.

1. Klassen DisjointSets

Du finner på itslearning filen *DisjointSets.h* i vilken deklarationen av klassen DisjointSets är gjord enligt nedan:

```
class DisjointSets
{
private:
    int* set; // array för att representera mängderna
    int size;  // storleken på arrayen
    // hjälpfunktioner efter behov
public:
    DisjointSets(int size); // size är antalet mängder
    DisjointSets(const DisjointSets &orig);
    virtual ~DisjointSets();
    DisjointSets& operator=(const DisjointSets &orig);
    int find(int x) const;
    int findCompress(int x);
    void unionSets(int root1, int root2);
    void unionSetsRank(int root1, int root2);
    int maxHeight() const; // bestämmer höjden på det högsta trädet
};
```

Du ska definiera konstruktor, destruktor samt medlemsfunktioner för denna klass i en fil namngiven *DisjointSets.cpp*. Initialt är antalet disjunkta mängder `size`. Vad funktionerna ska utföra framgår av namngivning på dem samt i vissa fall en kort kommentar.

2. Effekt av olika kombinationer av heuristiker för find och union

I en annan fil innehållande main-funktionen implementerar du testet. Här skapar du 4 objekt av typen DisjointSets så att du har olika objekt för var och en av de kombinationer av find och union som ska undersökas. Definiera en konstant för antalet element (var och ett i en egen mängd vid initialiseringen) vilken sätts till 10000.

Operationen union ska utföras 5000 gånger där mängderna slumpas fram. Det måste vara identiska mängder för alla 4 objekten.

Testerna ska upprepas 100 gånger och medelvärdet av maxhöjden på träden för respektive DisjontSet-objekt ska beräknas (använd funktionen maxHeight()). Resultaten ska presenteras i en tabell med 2 rader (**union** utan och med rank) och 2 kolumner (**find** utan och med komprimering).

Tips! Gör union på returerna av find(...) resp findCompress(...) då dessa visat att två disjunkta mängder identifierats.

3. Användningsområden för disjunkta mängder

Disjunkta mängder kan bl. användas för att åstadkomma labyrinter från ett rutnät men det finns även andra användningsområden för disjunkta mängder. Beskriv hur disjunkta mängder används för att konstruera labyrinter samt ange ytterligare 2 användningsområden och beskriv hur disjunkta mängder användas i dessa båda fall.

4. Rapport

Du ska skriva en rapport som lämnas in i pdf-format med minst följande innehåll:

- En introduktion (3 – 10 rader) där du med egna ord beskriver vad ADT Disjunkta mängder är samt principen de olika heuristikerna för find och union.
- Kommentarer på testresultaten (punkt 2) där du reflekterar över hur heuristikerna påverkar höjden på de träd som representerar mängderna.

  Variera antalet unioner (behåll antalet mängder) och försök finna någon situation där samtidig användning av båda heuristikerna ger en klar förbättring.
- Redovisning av användningsområden för disjunkta mängder (punkt 3 ovan)
- Tidsredovisning av projektet där du anger tid för inläsning, design, implementation, testning och rapportskrivning.

5. Slutprodukt och inlämning:

Slutprodukten är klassen DisjointSets tillsammans med program som visar testresultaten av de olika kombinationerna av operationerna find och union samt en rapport enligt punkt 4 ovan.

Du lämnar de C++ filer som ingår i din lösning samt rapporten (pdf-fil) som en packad fil vilken du namnger med ditt förnamn och efternamn följt av ProjectDisjointSets.