# Projekt i DV1490 våren 2017

*Projektet görs individuellt eller i grupp om två studenter. Projektrapporten ska skrivas individuellt. (Se kursplanen.)*

*Det måste tydligt framgå av rapportens framsida om arbetet gjorts individuellt eller i samarbete med en annan namngiven student.*

*Om arbetet gjorts i grupp om två ska den individuellt skrivna rapporten innehålla ett avsnitt om hur arbetsfördelningen gjorts. Du ska även kunna redogöra för hela projektet om du får frågor på arbetet.*

## 0. Inledning

En abstrakt datatyp för **disjunkta mängder** beskrivs i kapitel 8 av kursboken *Data Structures and Algorithm Analysis in C++*. I projektet ska du göra följande.

a) Implementera ADT **Disjunct Sets** med hjälp av träd så som beskrivs i kursboken 8.3. Använd *DisjointSets.h* som finns på Itslearning (se **punkt 1** nedan).

b) Skriva ett program som genom tester visar effekten av två olika strategier för **union** (boken 8.4) samt utan och med heuristiken stigkomprimering (*path compression,* boken 8.5) för operationen **find**. Se **punkt 2** nedan.

c) Skriva en kort rapport enligt instruktioner i **punkt 3** längre ned.

## 1. Klassen DisjointSets

På Itslearning finns filen *DisjointSets.h* som deklarerar klassen DisjointSets. Du ska definiera konstruktor, destruktor samt medlemsfunktioner för denna klass i en fil *DisjointSets.cpp* där `size` anger antalet element (= antalet mängder).

## 2. Tester med olika kombinationer av heuristiker för find och union

I en annan fil innehållande main-funktionen implementerar du testerna. Här skapar du fyra objekt av typen DisjointSets; ett objekt för var och en av de kombinationer av **find** och **union** som ska undersökas. Definiera en konstant för antalet element vilken sätts till 10000. (Varje element ska läggas i en egen mängd vid initialiseringen.)

Operationen **find** ska utföras 5000 gånger för element som slumpas fram. Operationen **union** ska också göras 5000 gånger med slumpmässigt valda element. Du väljer hur du ”blandar” dessa 10000 operationer (enklast är kanske att göra **find** varannan gång och **union** varannan gång). För att få jämförbara testresultat ska varje slumpad serie användas för de fyra objekten (dvs, de fyra olika kombinationerna av heuristiker för **find** och **union**).

Givetvis är det enbart disjunkta mängder som ska ”slås ihop” med **union**. Använd värden returnerade av find(…) respektive findCompress(…) för att undersöka detta.

Som mått på ”tid” ska ditt program räkna totala antalet trädnoder som besöks av find(…) respektive findCompress(…). Använd medlemsvariabeln `count_steps` för detta.

Testerna ska upprepas 100 gånger. Medelvärdet av de hundra testernas antal besökta noder för respektive DisjontSet objekt ska beräknas. Resultaten ska presenteras i en tabell med två rader (**union** utan och med rank) och två kolumner (**find** utan och med stigkomprimering). Även standardavvikelsen ska beräknas och presenteras i en tabell.

## 3. Rapport

Du ska skriva en rapport som lämnas som pdf fil. Minimumkrav är att rapporten innehåller följande.

- En introduktion (3 – 10 rader) där du med egna ord beskriver vad ADT **Disjunkta mängder** är samt principen för de olika heuristikerna för **union** och **find**.

- Det finns flera användningsområden för disjunkta mängder. Beskriv hur disjunkta mängder kan används för att konstruera labyrinter. Ange ytterligare två användningsområden och beskriv hur disjunkta mängder används i dessa fall.

- Testresultaten i tabellform.

- Kommentarer till testresultaten där du reflekterar över hur olika val av heuristiker påverkar effektiviteten.

  Försök köra testerna med större antal element och fler **find**/**union** operationer om användning av båda heuristikerna samtidigt inte gett någon märkbar förbättring. (Programmet med testerna ska givetvis inte ta orimligt lång tid köra.)

- Inkludera en tidsredovisning av projektet där du anger tid för inläsning, design, implementation, testning samt rapportskrivning.

- Om programmeringen gjorts i grupp om två ska den individuellt skrivna rapporten innehålla ett avsnitt om hur arbetsfördelningen varit och det ska på rapportens framsida framgå vem som var samarbetspartner.

## 4. Slutprodukt och inlämning

Slutprodukten är klassen DisjointSets tillsammans med programmet som testar de olika kombinationerna av operationerna **find** och **union** samt en rapport enligt **punkt 3** ovan.

Du lämnar de C++ filer som ingår i din lösning samt rapporten (som pdf) i en packad fil. Filnamnet ska vara ditt för- och efternamn följt av ProjectDisjointSets.