Derde programmeeropgave — Dynamisch programmeren
Algoritmiek 2010, Universiteit Leiden

Het probleem

In de plaats Nieuw-Amsterdam staan vrijwel alleen maar zeer hoge torenflats. Krantenbezorgers die alle abonnees in zo'n flatgebouw hun ochtendkrant moeten bezorgen willen dit uiteraard zo snel mogelijk doen, zodat ze niet al te vroeg op hoeven te staan. In deze opgave zullen we berekenen wat het minimale aantal stappen is dat een krantenbezorger moet doen om alle kranten bij de abonnees in een gegeven flat op de juiste wijze te bezorgen. De invoerfile zal een aantal flats bevatten, met daarin aangegeven de posities van de abonnees, waarvoor het optimale aantal stappen moet worden berekend.

Een krantenbezorger begint altijd op de begane grond van de flat en gaat vervolgens gestaag naar boven, daarbij per verdieping alle kranten bezorgend. Van elke flat is het aantal verdiepingen en het aantal woningen per verdieping bekend, alsmede de positie van de ingang op de begane grond. (De flat heeft precies één ingang.) Eveneens zijn de locaties van de abonnees bekend. In elk flatgebouw zit zowel helemaal links als helemaal rechts een trap. Beide trappen lopen van de begane grond tot de bovenste verdieping. De krantenbezorger zal zo weinig mogelijk trappen willen beklimmen, hetgeen inhoudt dat hij altijd eerst bij alle abonnees op een verdieping de kranten bezorgt voordat hij naar de volgende verdieping gaat. De bedoeling is om voor elk van een aantal gegeven flats te bepalen hoeveel tijd (stappen) het minimaal kost om alle kranten te bezorgen. Bovendien dient vervolgens de/een bijbehorende optimale bezorgroute gegeven te worden voor elke flat.

De opdracht

De opdracht bestaat uit twee delen.

1. Een **C++-programma** dat het volgende doet.

   Specificaties en opmerkingen:

   - Er moet een programma worden geschreven dat gebruikmaakt van *bottom up dynamisch programmeren*.
   - De gebouwen en hun abonnees worden ingelezen uit een invoerfile.
   - De eerste regel van de invoer bevat een geheel getal `flats`, het aantal flatgebouwen ($1 \leq$ `flats` ≤ 100). Daarna volgen de gegevens van de flatgebouwen zelf.
   - Elke flat wordt beschreven door een regel met twee integers, die het aantal verdiepingen (`hoogte`) resp. de breedte `breedte` van de flat aangeven. De breedte geeft dus per verdieping het aantal woningen + twee trappen aan. De linkertrap bevindt zich op positie 0, de rechtertrap op positie `breedte`-1 en de woningen zelf op plekken 1 t/m `breedte`-2. Er geldt verder: $1 \leq$ `hoogte` ≤ 30 en $4 \leq$ `breedte` ≤ 80.
     Vervolgens komen er `hoogte`$+1$ regels die de layout van de flat beschrijven, en wel als volgt. De eerste regel beschrijft het dak van de flat: een '+' gevolgd door `breedte`-2

mintekens, afgesloten door weer een '+'. Elke volgende regel beschrijft een verdieping (van hoog naar laag) en begint en eindigt met een 'T' (trap). Daartussenin bevinden zich de woningen, waarbij 'Y' een woning aangeeft waar een krant moet worden bezorgd, en 'N' een woning waar geen krant moet worden bezorgd.
Op de begane grond bevinden zich behalve twee trappen en woningen ook de ingang tot de flat; deze wordt aangeduid met het karakter 'I'.

- Een voorbeeldbeschrijving van een flat:

```
5 12
+----------+
TNNNYNNYNNYT          // vierde verdieping (= bovenste verdieping)
TNNNNNNNNNNT          // derde verdieping
TYNNYYYYYYYT          // tweede verdieping
TYNYNYYYNYNT          // eerste verdieping
TNNNINYNNNYT          // begane grond met ingang op positie 4 (trap op plek 0)
```

- Elke verdieping heeft tussen de nul en `breedte`-2 abonnees; echter op de bovenste verdieping is er altijd ten minste één.
- Het bezorgen van een krant komt gewoon neer op langs de woning van een abonnee gaan. Het passeren van een woning kost precies één stap; het beklimmen van een trap telt ook voor één stap. Het tellen van het aantal stappen stopt zodra de bezorger bij de laatste abonnee op de bovenste verdieping is gekomen.
- Voorbeeldinvoer met bijbehorende minimale aantal benodigde stappen:

```
2
5 12
+----------+
TNNNYNNYNNYT
TNNNNNNNNNNT
TYNNYYYYYYYT
TYNYNYYYNYNT
TNNNINYNNNYT
10 10
+--------+
TNNNYYNNNT
TYYNNNNNYT
TNYNNNNYYT
TNYYNNNYNT
TNNYNNYYNT
TNNYYNYNNT
TNNNYYYNNT
TNNNYYNNNT
TNNNNYNNYT
TINNNNNNYT
```

Voor de eerste flat: minimaal 40 stappen; voor de tweede flat: minimaal 93 stappen.

- Ook moet een optimale bezorgroute worden opgeleverd: hiervoor hoef je alleen aan te geven via welke trappen (links of rechts) je achtereenvolgens van de ene naar welke volgende verdieping gaat.
  Voor het eerste voorbeeld in bovenstaande voorbeeldinvoer: trap rechts van 0 naar 1; trap links van 1 naar 2; trap rechts van 2 naar 4. Hier zijn geen abonnees op verdieping 3, dus die verdieping sla je over.
  In het tweede voorbeeld begint de optimale route met: trap rechts van 0 naar 1; trap rechts van 1 naar 2;. Op verdieping 1 keer je om zodra je de laatste abonnee aldaar de krant hebt bezorgd.
- Je kunt volstaan met één klasse en bijbehorende memberfuncties.
- Boven elke gebruikte functie moet een commmentaarblokje komen met daarin een (zeer) korte beschrijving van wat de functie doet. Noem daarin ook de gebruikte parameters: geef hun betekenis en geef aan hoe ze eventueel veranderd worden door de functie. Geef bij memberfuncties ook aan wat deze met de membervariabelen van het object doen. Let ook op de layout (consequent inspringen) en op het overige commentaar bij de programmacode (zinvol en kort).
- Het programma moet (ook) onder UNIX draaien.

2. Een getypt **verslag** in LaTex, waarin eerst kort het problem wordt uitgelegd, en vervolgens duidelijk wordt toegelicht hoe je het hebt oplost. In die toelichting staat in elk geval

   - wat voor tabel gebruikt wordt voor het opslaan van tussenresultaten
   - de recurrente betrekking / recursieve formulering die gebruikt wordt om de tabel te vullen (met uitleg hoe je daaraan komt)
   - welke berekeningsvolgorde je gebruikt en waarom
   - hoe je uit de tabel de/een optimale bezorgroute haalt.

Eventuele extra informatie over de programmeeropdracht wordt gegeven tijdens het college en/of is te vinden op: `http://www.liacs.nl/~mwitsenb/Algoritmiek`
Voor meer informatie over het vak, laatste wijzigingen, etcetera, raadplege men:
`http://www.liacs.nl/~graaf/ALGO/`

Uiterste inleverdatum: dinsdag 18 mei 2010. Voor elke week te laat inleveren gaat er een punt van het cijfer af. Het programma per e-mail sturen aan de hoofdnakijker: tijn@liacs.nl. Listing en verslag moeten op papier worden ingeleverd en in de daartoe bestemde doos met opschrift Algoritmiek in de postkamer van Informatica (kamer 156) worden gedeponeerd of persoonlijk overhandigd. Vermeld overal duidelijk de namen van de makers.

Normering: verslag 2; commentaar en layout 1,5; modulaire opbouw en OOP 1,5; werking 5.