Derde programmeeropgave — Dynamisch programmeren
Algoritmiek voorjaar 2008, Universiteit Leiden

Het probleem

Gegeven zijn twee rijen karakters (strings) $X = < x_1, x_2, \ldots, x_m >$ en $Y = < y_1, y_2, \ldots, y_n >$ (m en n geheel en ≥ 0). Gevraagd wordt de/een langste gemeenschappelijke deelrij van X en Y.

Definitie 1
Een rij $Z = < z_1, z_2, \ldots, z_k >$ is een deelrij van $X = < x_1, x_2, \ldots, x_m >$ als Z uit X verkregen kan worden door daaruit elementen weg te laten. De volgorde van de elementen in Z is dus als in X. Zo is $< C, D, A >$ wel een deelrij van $< A, B, C, B, D, A, B >$, maar $< D, C, A >$ niet.
Definitie 2
Een rij $Z = < z_1, z_2, \ldots, z_k >$ is een gemeenschappelijke deelrij van $X = < x_1, x_2, \ldots, x_m >$ en $Y = < y_1, y_2, \ldots, y_n >$ als Z een deelrij is van zowel X als Y.

Voorbeeld
Als $X = < A, B, C, B, D, A, B >$ en $Y = < B, D, C, A, B, A >$, dan is $< B, C, A >$ een gemeenschappelijke deelrij (ter lengte 3) van X en Y. Het is echter niet een *langste* gemeenschappelijke deelrij, want $< B, C, B, A >$ is een deelrij van zowel X als Y, ter lengte 4. Het is zelfs een langste gemeenschappelijke deelrij (LGD) van X en Y, aangezien er geen gemeenschappelijke deelrijen ter lengte 5 bestaan. De rij $< B, D, A, B >$ is overigens ook een LGD van X en Y.

De opdracht

Het probleem is dus om gegeven twee rijen, X en Y, een langste gemeenschappelijke deelrij te vinden. De bedoeling is om dit zowel via top down dynamisch programmeren (= recursief met gebruik van een hulparray om tussenresultaten in op te slaan) op te lossen, als via bottom up dynamisch programmeren.
De opdracht bestaat uit twee delen: een **C++-programma** en een **verslag**.

Opmerkingen en aanwijzingen

1. Het programma moet gebruik maken van *dynamisch programmeren* (DP). De gebruiker moet kunnen kiezen tussen top down en bottom up DP.
2. Er moet ingelezen worden uit een invoerfile. Die invoerfile bevat allereerst een geheel getal n dat het aantal testgevallen aangeeft. Op de n volgende regels staan telkens twee rijen karakters (de X en Y waarvan de LGD moet worden bepaald), gescheiden door een spatie. Je mag aannemen dat de invoerfile correct is, dus voldoet aan de specificaties.
3. Een oplossing van het probleem, een langste gemeenschappelijke deelrij van twee rijen, $< x_1, x_2, \ldots, x_m >$ en $< y_1, y_2, \ldots, y_n >$, moet worden uitgedrukt in oplossingen van deelproblemen. In dit geval zijn dat LGD's van prefixen van X en Y, dus rijtjes van de vorm $< x_1, x_2, \ldots, x_i >$ en $< y_1, y_2, \ldots, y_j >$ (met $i < m$ en $j < n$). Leg in het verslag uit hoe dat moet en laat zien waarom het klopt.
4. Leg uit, bijvoorbeeld met een voorbeeld, dat bij een recursieve oplossing sprake is van een watervaleffect, en dat dus dynamisch programmeren gewenst is.

5. Er moeten twee functies geschreven worden: een die voor een gegeven X en Y via top down DP de *lengte* van een optimale oplossing bepaalt, en een die dat doet via bottom up DP. Definieer hiertoe een geschikte tabel.
6. Vervolgens moet uit de tabel ook een optimale oplossing worden afgeleid. Er moet dus een langste gemeenschappelijke deelrij worden gegenereerd. Bij de top down methode moet dat gedaan worden door de tabel te gebruiken en terug te redeneren. Bij de bottom up methode moet een geschikte tweede tabel worden gebruikt, waarvan de entries tegelijk met het vullen van de eerste tabel bepaald wordt. Vergelijk het busreisprobleem van college.
7. De gebruiker kan kiezen of de testgevallen uit de invoerfile allemaal top down "behandeld" worden of allemaal "bottom up". Voor elk testgeval moet dan een optimale oplossing worden gegenereerd en (steeds op een nieuwe regel) naar een uitvoerfile worden geschreven.
8. Je kunt volstaan met één klasse en bijbehorende memberfuncties.
9. 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).
10. Het programma moet (ook) onder UNIX draaien.
11. Het verslag moet getypt zijn en bevat een inleiding waarin kort het probleem wordt uitgelegd, en een duidelijke toelichting op hoe het wordt opgelost. In die toelichting staat in elk geval:
    - de twee eerder genoemde punten (recursieve formulering van de optimale oplossing en watervaleffect).
    - wat voor tabel gebruikt wordt voor het opslaan van tussenresultaten en de recurrente betrekking die gebruikt wordt om de tabel te vullen (met uitleg)
    - welke berekeningsvolgorde je gebruikt bij het vullen van de tabel in het bottom up geval en waarom
    - hoe je uit de tabel de/een optimale oplossing zelf haalt, en dit zowel voor de top down methode als de bottom up methode
    - welke van de twee DP-methodes sneller is (of misschien blijken ze wel even snel te zijn)

Eventuele extra informatie over de programmeeropdracht wordt gegeven tijdens het college en/of is te vinden op: `http://www.liacs.nl/home/jlaros/edu/alg`
Voor meer informatie over het vak, laatste wijzigingen, etcetera, raadplege men:
`http://www.liacs.nl/home/graaf/ALGO/algo2008.html`
Uiterste inleverdatum: maandag 19 mei 2008. Voor elke week te laat inleveren gaat er een punt van het cijfer af. Het programma per e-mail sturen aan de hoofdnakijker: jlaros@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 2; modulaire opbouw en OOP 1; werking 5.