Tentamen Algoritmiek vrijdag 30 januari 2004

Deelnemers hadden 3 uur de tijd voor dit tentamen. Bij elk onderdeel staat het aantal punten dat gehaald kon worden.

Schrijf duidelijk en helder. Wanneer U een gedeelte van een opgave niet heeft kunnen aantonen, dan mag U dat resultaat toch gebruiken in een later gedeelte van die opgave. (Dit is met name van toepassing voor opgave 5.)

1 Maximum stroming in netwerken

(i) (1 punt) Leg in eigen woorden *duidelijk* uit hoe het Ford-Fulkerson algoritme werkt. (U hoeft *niet* uit te leggen waarom het algoritme correct werkt, maar moet *wel* duidelijk aangeven welke stappen het algoritme doet, en hoe dit soort stappen er uit zien.)

(ii) (1 punt) Zou U bij een implementatie van het Ford-Fulkerson algoritme de voorkeur geven aan het gebruik van een representatie van het netwerk met een *adjacency list* of een *adjacency matrix* representatie? Waarom?

2 Cycles in grafen

(1.2 punt) Geef een algoritme, dat gegeven een gerichte graaf $G = (N, A)$, als output een knoop v oplevert die op een cycle in G ligt, of antwoordt dat G geen enkele cycle bevat. Uw algoritme moet in $O(n + a)$ tijd werken.

3 Dynamisch programmeren: Chained Matrix Multiplication

We hebben een serie matrices van verschillend formaat, $A_1, \ldots, A_r$. Matrix A_i is van formaat d_{i-1} bij d_i. We willen het product van de matrices uitrekenen: $A_i \cdot A_2 \cdot A_3 \cdots A_{r-1} \cdot A_r$. Echter, in plaats van het 'klassieke' algoritme voor het vermenigvuldigen van twee matrices, gebruiken we de volgende methode: we vergelijken wat sneller is: de klassieke matrixvermenigvuldiging, of het opvullen van de matrices tot vierkante matrices, dan Strassen's algoritme gebruiken, en dan de niet-relevante posities weer weglaten.

Gegeven is een constance c_0, zodat geldt dat het vermenigvuldigen van een matrix van formaat k_0 bij k_1 met een matrix van formaat k_1 bij k_2 in het totaal $c_0 \cdot \min(k_0 \cdot k_1 \cdot k_2, (\max\{k_0, k_1, k_2\})^{2.81})$ operaties kost.

Het probleem wat we bekijken is het bepalen van een volgorde van vermenigvuldingen (het 'haakjes' zetten) zodat het totaal aantal operaties zo klein mogelijk is. Bijvoorbeeld, als $r = 3$ kunnen we het product uitrekenen als $A_1 \cdot (A_2 \cdot A_3)$ of als $(A_1 \cdot A_2) \cdot A_3$. We gaan dit probleem aanpakken met behulp van dynamisch programmeren. In deze tentamenopgave beperken we ons tot het formuleren van de recurrente betrekking.

Definieer OPT(i,j) als het minimum aantal operaties nodig om $A_i \cdot A_{i+1} \cdots A_j$ te berekenen, $1 \leq i \leq j \leq r$.

(i) (0,2 punt) Zijn de volgende beweringen correct:

- Voor elke i, $1 \leq i \leq r$: OPT(i,i) $= 0$.
- Voor elke i, $1 \leq i \leq r-1$: OPT(i,$i+1$) $= c_0 \cdot \min(d_{i-1} \cdot d_i \cdot d_{i+1}, (\max\{d_{i-1}, d_i, d_{i+1}\})^{2.81})$.

Leg heel kort uit waarom wel of niet.

(ii) (1 punt) Geef een recurrente betrekking voor OPT(i,j).

4 Dynamisch programmeren II

(1 punt) Gegeven zijn twee strings, $X = x_1x_2 \cdots x_n$ en $Y = y_1y_2 \cdots y_m$. We willen de lengte van de langste gemeenschappelijk subsequence van X en Y vinden. Hiertoe definieren we voor $0 \leq i \leq n$ en $0 \leq j \leq m$:

$$M[i,j] \quad = \quad \text{de lengte van de langste gemeenschappelijke subsequence van } x_1x_2 \cdots x_i \text{ en } y_1y_2 \cdots y_j.$$

(Voor $i = 0$ is $x_1x_2 \cdots x_i$ per definitie de lege string, en voor $j = 0$ is $y_1y_2 \cdots y_j$ per definitie de lege string.) We kunnen de volgende recurrente betrekking opstellen voor $M[i,j]$:

$$M[i,j] = \begin{cases} M[i-1,j-1]+1 & \text{als } i>0,\ j>0, \text{ en } x_i = y_j \\ \max(M[i-1,j], M[i,j-1]) & \text{als } i>0,\ j>0, \text{ en } x_i \neq y_j \\ 0 & \text{als } i=0 \text{ of } j=0 \end{cases}$$

Geef een algoritme dat in $O(nm)$ tijd de lengte van de langste gemeenschappelijke subsequence van X en Y berekent. (N.B. U hoeft de recurrente betrekking hierboven dus niet aan te tonen; U mag wel aannemen dat deze geldig is.)

5 Een heuristiek voor het handelsreizigersprobleem

In deze opgave beschouwen we de *closest-point* heuristiek voor het handelsreizigersprobleem met driehoeksongelijkheid. We nemen ook aan dat de instantie symmetrisch is: voor elk paar steden v, w geldt $d(v,w) = d(w,v)$.

We beginnen met een 'triviale' cycle met slechts één willekeurig gekozen punt. Zolang de cycle nog niet alle punten bevat herhalen we de volgende stap. Stel C is de verzameling punten op de cycle. We zoeken een stad $u \notin C$, met $d(u,C) = \min\{d(u,w) |\ w \in C\}$ zo klein mogelijk. Met andere woorden, van alle punten die nog niet op de cycle liggen kiezen we diegene die zo dicht

mogelijk bij een punt op de cycle ligt. Laat w het punt op de cycle zijn dat het dichtst bij u ligt. (D.w.z.: $d(u,w) = \min_{x \notin C, y \in C} d(x,y)$.) Voeg nu u toe aan de cycle, direct na w. De knoop die eerst na u komt, komt dus nu na w op de cycle. We nemen aan dat de driehoeksongelijkheid geldt.

(i) (0.2 punt) Formuleer de driehoeksongelijkheid voor het handelsreizigersprobleem.

(ii) (0.6 punt) Beargumenteer dat wanneer u toegevoegd wordt, de lengte van de cycle met hooguit $2 \cdot d(u,w)$ toeneemt.

(iii) (0.7 punt) Kijk naar de volgende verzameling kanten F. Initieel is F leeg. Wanneer we u toevoegen met dichtstbijzijnde knoop w op de cycle, dan stoppen we de kant $\{u,w\}$ in de verzameling F. Laat F^* de uiteindelijke verzameling kanten F zijn nadat alle knopen zijn toegevoegd aan de cycle. Beargumenteer dat F^* een minimum opspannende boom is van de graaf gevormd door de handelsreizigersinstantie.

(iv) (0.4 punt) Beargumenteer dat de lengte van een optimale tour groter of gelijk is aan de som van de lengtes van alle kanten in F^*.

(v) (0.4 punt) Toon aan dat de closest-point heuristiek een 1-relatief benaderingsalgoritme voor handelsreiziger met driekhoeksongelijkheid is.

6 Datastructuur voor disjuncte verzamelingen

(1 punt) We bekijken de datastructuur voor 'disjoint sets' (*'union-find'*) die gebruik maakt van bomen met union by rank en padcompressie. Leg kort uit wat de techniek van padcompressie inhoudt, en waarom die gebruikt wordt. (Maximaal 10 regels.)

7 Punten bedekken met intervallen

We bekijken het volgende probleem. Gegeven is een aantal n reële getallen $x_1, x_2, \ldots, x_n$, in stijgende volgorde, d.w.z., we hebben $x_1 < x_2 < \cdots x_n$. We zoeken een zo klein mogelijk aantal gesloten intervallen van precies 1 lengte elk die alle gegeven getallen bevatten.

(Een voorbeeld: als de getallen zijn 0,34 0,47 1,2 4.3 4.8 7.01, dan is [0,3 – 1,3], [4 – 5] en [6,7 – 7,7] een mogelijke optimale oplossing met drie intervallen.)

(i) (1 punt) Geef een algoritme dat dit probleem in $O(n)$ tijd oplost. Leg kort uit waarom uw algoritme correct werkt.

(ii) (0.3 punt) Welke algoritmische techniek(en) gebruikt Uw algoritme?