Tweede deeltoets Algoritmiek

15 april 2013, 9.00 – 11.00, Educ-γ.

Motiveer je antwoorden *kort*! Zet je mobiel uit. Stel geen vragen over deze toets; als je een vraag niet duidelijk vindt, schrijf dan op hoe je de vraag interpreteert en beantwoord de vraag zoals je hem begrijpt. Te halen 21pt, cijfer is totaal gedeeld door 2 (max 10).

1. **Routeplannen met BFS:** Een programmeur maakt een routeplanner voor in de auto en wil voor het zoeken van routes van A naar B een BFS doen met B als startpunt. De schil waar A in komt, de waarde van `d[A]`, geeft immers aan hoe ver A afligt van B, en de route is na afloop te vinden met de π-pointers in elke knoop. Waarom is dit geen goed idee? Hoe moet het wel?

2. **Complexiteiten:** Leg (kort!) het verschil uit tussen een verwachte, een gemiddelde, en een geamortiseerde complexiteit.

3. **Kortste Pad Algoritmen:** Wat is de worst-case complexiteit van Dijkstra's algoritme en van het Bellman-Ford algoritme voor single-source shortest path (op n knopen en m kanten)? Waardoor kun je in veel toepassingen, zoals routeplanners op een wegen-, spoorweg of communicatienetwerk, de daadwerkelijke performance van het Bellman-Ford algoritme veel beter krijgen? Onder welke omstandigheden kun je beter Dijkstra, en onder welke omstandigheden beter Bellman-Ford gebruiken?

4. **Kant kiezen bij MST:** In het Dijkstra/Prim algoritme voor Minimum Spanning Tree moet je telkens een *lichtste* kant uit een aantal kiezen en die toevoegen aan je tree. Wanneer er twee kanten van hetzelfde laagste gewicht zijn, die samen geen cykel introduceren, mag je er toch maar één daarvan toevoegen.

   (a) Geef hiervan een voorbeeld: een graaf met vijf knopen, waarbij er een *foute* MST ontstaat bij Dijkstra/Prim wanneer je toch beide lichtste kanten zou toevoegen.
   (b) Kun je bij Kruskal's algoritme wel meerdere kanten tegelijk toevoegen als die een gelijk minimaal gewicht hebben en geen cykel introduceren?

5. **Het Gat:** Gegeven is een gesorteerde array A van integers, waarbij $A[n-1] - A[0] \geq n$. Geef de invariant voor een algoritme dat een getal tussen $A[0]$ en $A[n-1]$ vindt, dat *niet* voorkomt in A. Geef ook het algoritme (het moet logaritmische tijd gebruiken).

6. **Max Flow in Polynomiale tijd:** Als je Ford-Fulkerson gebruikt met willekeurige paden (dus niet BFS) is de rekentijd begrensd door $O(m.|f^*|)$.

   (a) Waarom geldt dit *niet* als polynomiale tijd?
   (b) Waarom is het algoritme *wel* polynomiaal als je paden met BFS zoekt?

7. **NP:** Sommige mensen denken dat NP: *Niet-Polynomiaal* betekent, maar dat is niet zo. Wat betekent NP wel? Waarom is het geen goed idee om dit Niet-Polynomiaal te noemen?

8. **TSP met Closest Point:** Closest Point is een simpel algoritme voor TSP met driehoeksongelijkheid en symmetrie. Een cykel van steden wordt steeds uitgebreid. Begin met de triviale cykel die alleen het startpunt heeft. Telkens kies je het onbezochte punt dat het dichtst ligt bij een punt op de cykel. Als v het meest nabije nieuwe punt is, en u het cykelpunt waar v dichtbij ligt, stop dan v in de cykel na u. Herhaal tot alle punten in de cykel zitten.

   (a) Wat is het verschil tussen een *greedy algoritme*, een *heuristiek* en een *approximatie-algoritme*?
   (b) Welk van deze types is het Closest Point algoritme? Zeg kort waarom.