Advanced Algorithms and Datastructures - Exam Notes

André O. Andersen

2021

Max Flow

Disposition

Presentation

Linear Programming and Optimization

Disposition

Presentation

Randomized Algorithms

Disposition

Presentation

Hashing

Disposition

Presentation

Van Emde Boas Trees

Disposition

Presentation

NP-Completeness

Disposition

Presentation

Exact Exponential Algorithms and Parameterized Complexity

Disposition

Presentation

Approximation Algorithms

Disposition

1. Introduction
2. Definition of the *approximation ratio*, a $\rho(n)$-*approximation algorithm* and a *randomized* $\rho(n)$-*approximation algorithm*.
3. The Vertex-cover problem
   (a) Introduction
   (b) Proof that `APPROX-VERTEX-COVER` is a 2-approximation algorithm
4. MAX-3-CNF
   (a) Introduction
   (b) Proof that the randomized algorithm for `MAX-3-CNF` is a randomized 8/7-approximation algorithm

Presentation

Definition of *approximation ratio*
We say that an algorithm for a problem has an ***approximation ratio*** of $\rho(n)$ if, for any input of size n, the cost C of the solution produced by the algorithm is within a factor of $\rho(n)$ of the cost C^* of an optimal solution

$$\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq \rho(n).$$

Definition of $\rho(n)$-*approximation algorithm*
If an algorithm achieves an approximation ratio of $\rho(n)$, we call it a $\rho(n)$-***approximation algorithm***.

Definition of *randomized* $\rho(n)$-*approximation algorithm*
We say that a randomized algorithm for a problem has an ***approximation ratio*** of $\rho(n)$ if, for any input of size n, the expected cost C of the solution procuded by the randomized algorithm is within a factor of $\rho(n)$ of the cost C^* of an optimal solution:

$$\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq \rho(n).$$

We call a randomized algorithm that achieves an approximation ratio of $\rho(n)$ a ***randomized* $\rho(n)$-*approximation algorithm***

Introduction to *vertex cover*
A ***vertex cover*** of an undirected graph $G = (V, E)$ is a subset $V' \subseteq V$ such that if (u, v) is an edge of G, then either $u \in V'$ or $v \in V'$ (or both). The size of a vertex cover is the number of vertices in it. The ***vertex-cover problem*** is to find a vertex cover of minimum size in a given undirected graph. We call such a vertex cover an ***optimal vertex cover***.

The set C of vertices that is returned by `APPROX-VERTEX-COVER` is a vertex cover, since the laogrithm loops until every edge in $G.E$ has been covered by some vertex in C.

Algorithm 1 APPROX-VERTEX-COVER

Require: Undirected graph G
1: $C = \emptyset$
2: $E' = G.E$
3: **while** $E' \neq \emptyset$ **do**
4: let (u, v) be an arbitrary edge of E'
5: $C = C \cup \{u, v\}$
6: remove from E' edge (u, v) and every edge incident on either u or v
7: **return** C

Proof that `APPROX-VERTEX-COVER` is a 2-approximation algorithm
Let A denote the set of edges that line 4 picked. Not two edges in A share

an endpoint. Thus no two edges in A are covered by the same vertex from an optimal cover C^*, and we have the lower bound

$$|C^*| \geq |A| \tag{1}$$

on the size of an optimal vertex cover. Since A consists of the edges between two vertices in C (and since all of the elements in C are unique), we have the (exact) upper bound on the size of the vertex cover returned

$$|C| = 2|A| \tag{2}$$

Combining equation (1) and (2), we obtain

$$|C| = 2|A| \leq 2|C^*|$$

Polygon Triangulation

Disposition

1. Introduction
2. The 3-coloring approach
   (a) Example on running the algorithm
   (b) Proving that the 3-coloring approach is optimal in worst case
3. Example on partitioning a polygon into monotone pieces + runtime analysis
4. Example on triangulating a monotone polygon + runtime analysis

Presentation