University of Tokyo: Advanced Algorithms **Summer 2010**

Lecture 1 — 8 April

Lecturer: Hiroshi Imai *Scribe: Christian Sommer*

The first lecture serves as an introduction to the course. Its content is based on [MR95, Chapter 1]. The first part consists of an analysis of a randomized version of Quicksort; the second part consists of an algorithm for the MinCut problem.

1.1 Randomized Quicksort

Sorting is a fundamental problem in computer science. Given a list of n elements of a set with a defined order relation, the objective is to output the elements in sorted order. Quicksort [Hoa62] is a particularly efficient algorithm that solves the sorting problem. We demonstrate how Quicksort works using an example.

Suppose Quicksort was given the task to sort the following 8 numbers.

$$2, 7, 6, 3, 1, 8, 5, 4$$

Quicksort first selects a *pivot* element, say $\boxed{3}$. This pivot element is compared with all the elements on both sides, dividing the list of elements into two lists.

$$2, 7, 6, \boxed{3}, 1, 8, 5, 4$$

After comparisons with the pivot, the remaining elements are divided into two parts: one that consists of the elements smaller than the pivot, and one that consists of the elements larger than the pivot.

$$\underbrace{2, 1,}_{<3} \boxed{3}, \underbrace{6, 7, 8, 5, 4}_{>3}$$

We recurse on both parts.

$$\underbrace{2, 1,} \boxed{3}, \quad \underbrace{6, 7, 8, 5, 4}$$
$$\underbrace{\boxed{2}, 1}_{\dots} \quad \underbrace{6, 7, \boxed{8}, 5, 4}$$
$$\underbrace{6, 7, 5, 4,} \quad \boxed{8}$$
$$\underbrace{6, 7, \boxed{5}, 4,}$$
$$\underbrace{4}, \boxed{5}, \underbrace{7, 6}_{\dots}$$

Although the Quicksort algorithm is very efficient in practice, its worst-case running time is rather slow. When sorting n elements, the number of comparisons may be $O(n^2)$. The worst case happens if the sizes of the subproblems are not balanced. The selection of the pivot element determines the sizes of the subproblems. It is thus crucial to select a 'good' pivot.

In the following, we prove that if the pivot is selected *uniformly at random*, the expected number of comparisons of this randomized version of Quicksort is bounded by $O(n \log n)$.

Let $a = (a_1, a_2, \ldots a_i, \ldots a_j, \ldots a_n)$ denote the list of elements we want to sort, *in sorted order*. Note that, *for the analysis* we may assume that we know the order. The input is any permutation of a. For all $1 \leqslant i < j \leqslant n$, let $X_{i,j}$ denote the random variable indicating whether Quicksort compared a_i and a_j.

$$X_{i,j} = \begin{cases} 1 & \text{if Quicksort compares } a_i \text{ and } a_j \\ 0 & \text{otherwise} \end{cases}$$

Any two elements get compared at most once. The expected number of comparisons is thus

$$E\left[\sum_{i=1}^{n} \sum_{j=i+1}^{n} X_{i,j}\right] = E\left[\sum_{1 \leqslant i < j \leqslant n} X_{i,j}\right] = \sum_{1 \leqslant i < j \leqslant n} E[X_{i,j}]$$

Quicksort compares a_i and a_j if and only if either a_i or a_j is selected as pivot *before* any of the elements between a_i and a_j. Selecting an element between a_i and a_j as pivot will partition the list into two parts such that a_i and a_j lie in different parts and do not get compared. The number of elements between a_i and a_j is $j - i - 1$. Since in each step the pivot element is selected uniformly at random, the probability that either a_i or a_j is selected as pivot before any of the $j - i - 1$ elements between them is $2/(j - i + 1)$. If this happens, a_i and a_j are compared and $X_{i,j}$ is one. We thus have

$$E[X_{i,j}] = \frac{2}{j - i + 1}.$$

We derive

$$\begin{aligned} E\left[\sum_{i=1}^{n} \sum_{j=i+1}^{n} X_{i,j}\right] &= \sum_{i=1}^{n} \sum_{j=i+1}^{n} E[X_{i,j}] \\ &= \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{2}{j - i + 1} \\ &\leqslant 2 \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{1}{k} \\ &= 2n \sum_{k=1}^{n} \frac{1}{k} \\ &= O(n \ln n). \end{aligned}$$

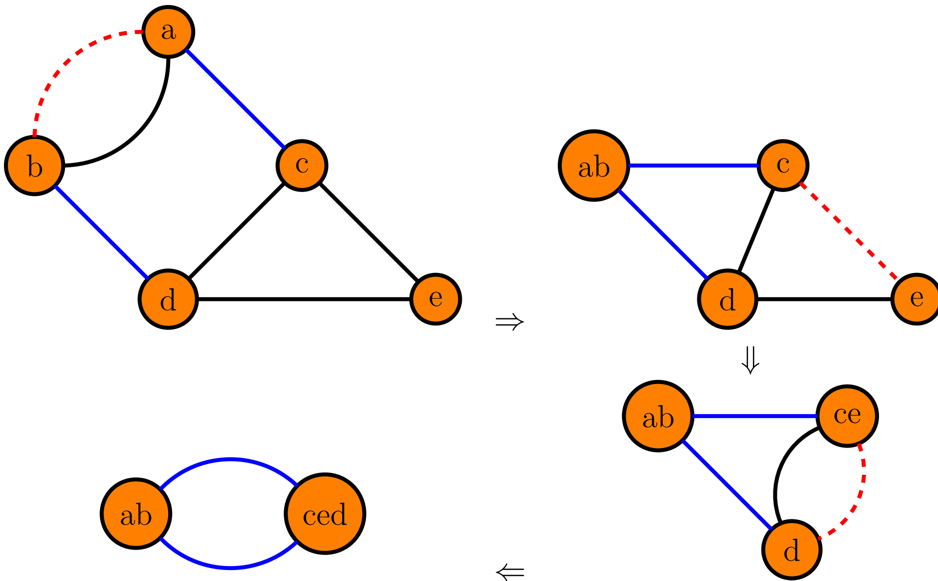

Figure 1.1. A sample execution of Algorithm 1 on a graph with 5 nodes. Throughout the execution, the edges of one min-cut of G are colored blue. At each step, the red, dashed edge is contracted. We are lucky: the blue min-cut "survives."

1.2 A Randomized MinCut Algorithm

The MinCut problem asks to partition the set of vertices of a (multi[1]-)graph $G = (V, E)$ into two non-empty, disjoint subsets $V_1 \cup V_2 = V$ such that the number of edges between V_1 and V_2 (meaning $|\{(v_1, v_2) \in E : v_1 \in V_1 \wedge v_2 \in V_2\}|$) is minimized.

Since the number of possible partitions grows exponential with n, an efficient algorithm cannot check all partitions. We analyze an efficient (expected polynomial-time) randomized algorithm due to Karger [Kar93]. Its pseudocode is listed as Algorithm 1. For an illustration, see Figure 1.1.

Algorithm 1 RandomizedMinCut($G = (V, E)$)

while the graph has more than two nodes **do**
  choose an edge $e = (u, v)$ uniformly at random
  contract e (the node that combines u and v "inherits" all the node labels of u and v) and remove self-loops {note that parallel edges are not removed}
end while
return the cut defined by the labels of one of the remaining nodes

At each iteration, the edge contraction step reduces the number of nodes by 1. The total number of iterations is thus $O(n)$.

Proposition 1.1. *Algorithm 1 returns a minimum cut with probability at least* $1/\binom{n}{2}$.

[1] A multi-graph is a graph that may have multiple "parallel" edges between pairs of nodes.

Proof: For $i \geqslant 0$, let $G_i = (V_i, E_i)$ denote the vertex-labelled multi-graph after i contractions. By definition, $G_0 = G$. Each node of G_i may serve as a representative for multiple nodes of G.

For $0 \leqslant j < i$, every cut of G_i corresponds to some cut of G_j. In particular, a min-cut of G_i also corresponds to a cut of G_j. The size of a min-cut of G_i is at least as big as the size of a min-cut of G_j.

Let us fix a min-cut of G. Let $C \subseteq E$ denote the set of edges of G that get cut by this min-cut. One of the main observations to prove the proposition is the following. If $c = |C|$, then each node must have degree at least c (otherwise there is a smaller min-cut that separates a node with degree less than c from the rest of the graph). This property implies $|E| \geqslant \frac{c}{2}n$.

Let the event S_i denote that C "survives" the contraction occurring on i nodes (contraction number $n-i+1$), which means that none of its edges was contracted. Let $\bar{S}_i$ denote its complement. We derive the probability that C survives contraction $i+1$ given that C already survived contraction i. We have the recursive relationship $Pr[S_{i+1}] = Pr[S_{i+1}|S_i] \cdot Pr[S_i]$.

$$\begin{aligned}
Pr[\bar{S}_n] &= \frac{|C|}{|E|} \leqslant \frac{c}{cn/2} = \frac{2}{n} \\
Pr[S_n] &= 1 - Pr[\bar{S}_1] \geqslant \frac{n-2}{n} \\
Pr[\bar{S}_i|S_{i+1}] &\leqslant \frac{c}{ci/2} = \frac{2}{i} \\
Pr[S_i|S_{i+1}] &\geqslant \frac{i-2}{i}
\end{aligned}$$

The probability that C survives the last contraction ($i = 3$) is thus at least

$$\begin{aligned}
Pr[S_3] &= Pr[S_3|S_4] \cdot Pr[S_4|S_5] \ldots Pr[S_{n-1}|S_n] \cdot Pr[S_n] \\
Pr[S_3] &\geqslant \frac{1}{3} \cdot \frac{2}{4} \cdots \frac{n-3}{n-1} \cdot \frac{n-2}{n} = \frac{(n-2)!}{n!/2} = \frac{2}{n(n-1)}.
\end{aligned}$$

This concludes the proof. □

The probability that Algorithm 1 returns a min-cut may seem small at first. However, suppose that we execute t independent[2] runs of Algorithm 1. Let $p = 1/\binom{n}{2}$ denote the "success" probability. The probability that all of the t runs "fail", meaning that the result is not a min-cut. The probability for this event is at most

$$(1-p)^t \leqslant e^{-tp}.$$

If we set the number of repetitions to $t \geqslant \binom{n}{2} \ln n$, the probability that none of the t independent runs succeeds is at most $1/n$.

[2]We use a new random seed for each execution.

Bibliography

[Hoa62] Charles Antony Richard Hoare. Quicksort. *Computer Journal*, 5(1):10–15, 1962.

[Kar93] David R. Karger. Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm. In *SODA*, pages 21–30, 1993.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge University Press, 1995.