Lecture 14 — Graph Contraction, Min Spanning Tree (DRAFT)

Parallel and Sequential Data Structures and Algorithms, 15-210 (Fall 2011)

Lectured by Guy Blelloch — October 13, 2011

Today:

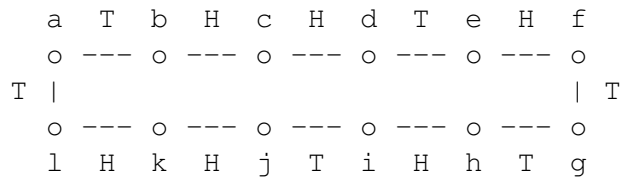
- Graph Contraction Continued
- Some Probability Theory

1 Graph Contraction Continued

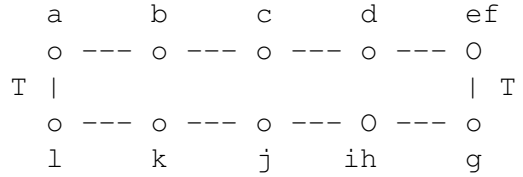
In the last class we were discussing the idea of graph contraction, which worked in a set of rounds each which contracts the graph. In each round we identify a set of disjoint connected subgraphs and contract each one into a single vertex, and then update the edges. Graph contraction is a powerful technique to generate parallel algorithms for a variety of problems on undirected graphs. The approach can also be applied to generate efficient algorithms on trees, although we won't cover these in this course.

To help understand the key issues in graph contraction we were considering a special case. In particular we were considering just edge contractions—where each connected subgraph we contract is an edge with its two endpoints. Furthermore we were just considering graphs that consists of a set of cycles. This might seem very limited, but actually it exposes most of what is interesting about graph contraction. After understanding this case, we will generalize to arbitrary graphs.

We had discussed how to select a set of independent edges in the graph (i.e. that don't share any endpoints) using randomization. In particular the rule we considered was flipping a coin on every edge and selecting an edge if it is heads and its two neighbors in the cycle are tails. This can easily be done in parallel and gives a probability of $1/8$ that each edge is selected. For example, in the following cycle with the indicated coin flips we will select 2 out of the 12 edges.



After contracting these two edges we will have 10 edges remaining for the next round.



Using this scheme we remove an average of $1/8^{th}$ of the remaining edges on each round. Assuming we always remove exactly $1/8^{th}$ we would have the following recurrence for the number of rounds

$$R(n) = R(7n/8) + 1.$$

Since the size is decreasing geometrically this solves to $O(\log n)$ rounds. Furthermore if we use array sequences to represent the graph, each round can be done in $O(\log n)$ span (needed for filtering out the remaining edges), and $O(n)$ work. This would give $O(\log^2 n)$ overall span, and the recurrence for work is

$$W(n) = W(7n/8) + O(n) = O(n).$$

Unfortunately we don't remove exactly $1/8^{th}$ of the vertices, but the number removed varies from round to round depending on the toss of the coins. We therefore need to be more careful about the analysis. For this purpose probability theory is very useful. We therefore will spend some time here reviewing basic definitions as well as introducing some new ideas you probably have not seen. These ideas will also be useful elsewhere in the course.

Probability Theory: Review Recall from probability theory that a *sample space* is the set Ω of all possible outcomes. In the case of flipping coins on n edges, the size of the sample space is 2^n representing all possible outcomes, which are all equally likely if the coins are unbiased. An *event* $\mathcal{E} \subseteq \Omega$ is any subset of the sample space. In our example, it might consist of all outcomes such that a particular edge is selected (it comes up heads and its neighbors tails), or all outcomes where the number of edges selected is at least $7n/8$. We can then ask what the probability of that event is by simply counting the number of outcomes in the event and dividing by the total number of possible outcomes, i.e., $\mathbf{Pr}[\mathcal{E}] = |\mathcal{E}|/|\Omega|$ if each outcome in the sample space is equally likely to happen.

A *random variable* is a function $f : \Omega \to \mathbb{R}$. We typically denote random variables by capital letters $X, Y, \ldots$. That is, a random variable assigns a numerical value to an outcome $\omega \in \Omega$. In our algorithm, one useful random variable is the count of number of selected edges. It maps each outcome to a number between 0 and $n/2$ (why not n?). Another is simply the count on a single edge which maps an edge to either 1 (if selected) or 0 (if not selected). Such a random variable that is zero or 1 is sometimes called an *random indicator variable*. For an event $\mathcal{E}$, the indicator random variable $\mathbb{I}\{\mathcal{E}\}$ takes on the value 1 if $\mathcal{E}$ occurs and 0 otherwise.

The *expectation* (or expected value) of a random variable is simply the weighted average of the value of the function over all outcomes. The weight is the probability of each outcome, giving

$$\mathbf{E}[X] = \sum_{\omega \in \Omega} X(\omega)\,\mathbf{Pr}[\omega]$$

in the discrete case. Applying this definition, we have $\mathbf{E}[\mathbb{I}\{\mathcal{E}\}] = \mathbf{Pr}[\mathcal{E}]$.

In our example, we are interested in the expectation that each single edge is selected and the expectation on the total number of edges selected. The expectation that a single edge is selected is $1/8$ since there are 8 possible outcomes on the coin tosses of three neighbors, and only one of them leads to the edge being selected.

Recall that one of the most important rules of probability is *linearity of expectations*. It says that given two random variables X and Y, $\mathbf{E}[X] + \mathbf{E}[Y] = \mathbf{E}[X + Y]$. This does not require that the

events are independent. In particular, if the events are selection of edges, two neighboring edges are certainly not independent (why?), but yet we can still add their expectations. So the expectation that two neighboring edges are selected is $1/8 + 1/8 = 1/4$. And hence the expected total number of edges selected is $n \times 1/8 = n/8$.

The *union bound* (Boole's inequality) states that for a set of events $A_1, A_2, \ldots$, that

$$\mathbf{Pr}\left[\bigcup_i A_i\right] \leq \sum_i \mathbf{Pr}\left[A_i\right].$$

If the events do not overlap (their sets of outcomes do not intersect) then the equation is an equality since we just add up the probabilities of every outcome. If they do overlap then the probability of the union is strictly less than the sum of the probabilities since some outcomes are shared. We will find the union bound this useful in showing high-probability bounds, discussed below.

Now we discuss two important techniques we will use. The first is a way to solve for the expectation on recurrences when the size of a subcall is a random variable. This is exactly what we need for the contraction example. The second is the idea of high probability bounds and how they can be used to analyze span.

Recurrences with Random Variables We are interested in recurrences of the form $F(n) = F(n - X_n) + 1$ where X_n is a random variable with a distribution depending on n. In our example it is the number of edges we remove and hence $\mathbf{E}\left[X_n\right] = n/8$. However, we cannot simply plug in the $n/8$ to get $F(n) = F(n - n/8) + 1$ since X_n is a probability distribution and $n/8$ is just the average. But we can use the following useful lemma.

Lemma 1.1 (Karp-Upfal-Widgerson)**.** *Let $T(n) = 1 + T(n - X_n)$ where for each $n \in \mathbb{Z}_+$, X_n is an integer-valued random variable satisfying $0 \leq X_n \leq n - 1$ and $T(1) = 0$. Let $\mathbf{E}\left[X_n\right] \geq \mu(n)$ for all $n \geq 1$, where μ is a positive non-decreasing function of n. Then,*

$$\mathbf{E}\left[T(n)\right] \leq \int_1^n \frac{1}{\mu(t)} dt.$$

The proof is mostly mechanical, so we didn't do it in class. But we'll prove it here for completeness.

Proof of Lemma 1.1. Let

$$h(x) = \int_{t=1}^{x} \frac{dt}{\mu(t)}.$$

We'll show that $\mathbf{E}\left[T(n)\right] \leq h(n)$ by induction on n. The base case is trivial. Now assume that this holds for all $m < n$–and show that it holds for n. By the recurrence definition, we have

$$\begin{aligned}
\mathbf{E}\left[T(n)\right] &\leq 1 + \mathbf{E}\left[T(n - X_n)\right] \leq 1 + \mathbf{E}\left[h(n - X_n)\right] \\
&\leq 1 + \mathbf{E}\left[\int_{t=1}^{n-X_n} \frac{dt}{\mu(t)}\right] \leq 1 + \mathbf{E}\left[\int_{t=1}^{n} \frac{dt}{\mu(t)} - \int_{t=n-X_n}^{n} \frac{dt}{\mu(t)}\right] \\
&\leq 1 + h(n) - \mathbf{E}\left[\int_{t=n-X_n}^{n} \frac{dt}{\mu(n)}\right] \leq 1 + h(n) - \frac{\mathbf{E}\left[X_n\right]}{\mu(n)} \leq h(n).
\end{aligned}$$

□

Notice that we could change $h(x)$ to $\sum_{i=1}^{x} \frac{1}{\mu(i)}$ and the proof will still go through. This gives us a slightly friendlier expression:

$$\mathbf{E}[T(n)] \leq \sum_{i=1}^{n} 1/\mu(i).$$

This lemma can be applied to many problems. In our case, it leads immediately to a solution for the expected number of rounds of contraction:

$$\begin{aligned} \mathbf{E}[R(n)] &\leq \int_1^n \frac{1}{t/8} dt \\ &= 8 \ln n. \end{aligned}$$

Therefore, the expected number of rounds is indeed logarithmic.

High Probability Bounds Unfortunately, knowing the expectation of components is not always good enough when composing costs. In analyzing the work of an algorithm (equivalent to time in sequential algorithms) we add the work for the various components. Linearity of expectations turns out to be very useful for this purpose since if we know the expected work for each component, we can just add them to get the overall expected work. However when analyzing span over a parallel computation we take the maximum of the span of the components. Unfortunately the following is (in general) not true: $\max(\mathbf{E}[X], \mathbf{E}[Y]) = \mathbf{E}[\max(X, Y)]$. We therefore cannot simply take the maximum of expectations across components to get the overall expected span.

As an example consider two parallel components each which have span either 1 or 5 each with .5 probability. The expected span for each of the components is therefore $(.5 \times 1 + .5 \times 5 = 3$. Now, in combination there are four possible configurations $((1,1),(1,5),(5,1),(5,5))$. Since we take the max of the two components the overall span for the four are $1, 5, 5, 5$. The expected span is the average of these: $.25 \times (1 + 5 + 5 + 5) = 4$. However if we simply took the maximum of the expected span of each component we would have $\max(3, 3) = 3$, which is wrong. The difference can be much greater than this example.

To deal with this issue instead (or in addition to) figuring out the expectation of a random variable we determine a high probability bound. We say that an event $\mathcal{E}$ takes place *with high probability* if

$$\mathbf{Pr}[\mathcal{E}] \geq 1 - \frac{1}{n^c}$$

for some constant $c \geq 1$. Here n is the instance size. Typically, in bounding the cost of an algorithm, we're interested in showing that $X_n \leq A$ with high probability. In other words, we want to say

$$\mathbf{Pr}[X_n \geq A] < \frac{1}{n^k}$$

This is saying that the probability of a "bad" event (the random variable takes on a value greater than some value A) is less than $1/n^k$ for some constant k. This probability is very low. We don't normally say something happens "with low probability", because we are interested in the opposite "good" case. For example, for us the X_n will be some cost measure and we want to say that with high probability the cost will not be greater than some A, i.e., that the probability that it is greater is small.

Combining high-probability bounds when calculating span turns out to be much easier than using expectation. We will get back to this. For now we will just state that it is possible to show that

$$\mathbf{Pr}\left[R(n) \geq 16k \ln n\right] < \frac{1}{n^k}.$$

So, the number of rounds of graph contraction assuming each rounds removes an expected constant fraction ($1/8$ in example) is $O(\log n)$ with high probability.

2 Graph Contraction on General Graphs

Now lets return to graph contraction but now considering general graphs instead of just cycles. Consider a star graph. A star graph is a graph with a single vertex in the middle and all the other vertices hanging off of it with no connections between them.

Draw picture.

Will edge contraction work well on this? How many edges can contract on each step.

Instead we will consider star contraction. The idea is to allow a vertex v to contract with all its neighbors. The vertex is called the star center. The neighbors can be connected with each other, but they are all viewed as contracting with v. We will use a coin to determine if the vertex will be the center of a star. The basic approach is:

```
Find stars:
  1.  Every vertex flips an unbiased coin to decide if it a center
  2.  Every non-center tries to hook up with a neighbor that is a center
  3.  All the hooks identify a set of stars
```

It should be clear that stars are disjoint since every non center only hooks up with one center, and the center does not hook up with anyone.

How many vertices do we expect to remove?