Lecture 7: Probabilistic Analysis

We finish the description of the divide and conquer algorithm for square matrix multiplication. See Chapter 28 for more details.

The main subject of this lecture is the hiring problem in Chapter 5. It is well explained in the textbook. So I will not repeat it here. The key points are:

- Indicator variables: binary variables associated with random events. An important property of indicator variable is its expected value is equal to the probability of the probability of the associated event. Moreover, indicator variables are commonly used to express a property of a larger system. For example, the number of total hires made is equal to the summation of the indicator variables, each defined for a particular candidate. This is a recurring situation in probabilistic analysis.

- Linearization of expectation: $E(X+Y) = E(X) + E(Y)$. This works even when $X$ and $Y$ are not independent. This is widely used, e.g. when indicator variables are used. See the hiring problem explanation in Chapter 5 for more details.

Lecture 8: More Probabilistic Analysis and Heapsort

We give a few more examples on probabilistic analysis. The first example concerns the situation where each of $n$ sailors takes arbitrary unoccupied cabin. We want to compute the expected number of sailors who sleep in their own cabins. We define an indicator variable $S_i$ for each sailor, where $S_i = 1$ if sailor $i$ sleeps in his own cabin. Then $S = \sum_{i=1}^{n} S_i$, and we want to compute $E(S)$. Note $E(S_i) = P($sailor $i$ sleeps in his own cabin$) = \frac{1}{n}$, since each sailor has the same chance sleeping in any cabin. By the linearity of expectation, we have $E(S) = \sum_{i=1}^{n} E(S_i) = 1$.

Our second example refers to the expected summation of two fair dices. Define $S_1$ and $S_2$ be the outcome of two throwing. We want to compute $E(S) = E(S_1 + S_2)$. One may use the definition of expected value and compute the probability of $S = i$, for each valid $i$. But this is tedious to compute. It is easier to use the fact $E(S) = E(S_1) + E(S_2)$. Here $E(S_1) = \sum_{v=1}^{6} v \cdot P(S_1 = v) = \sum_{v=1}^{6} v/6 = 3.5$. Thus, $E(S) = 3.5 + 3.5 = 7$.

The third example is the birthday paradox. See Section 5.4.1 for detailed explanation. I will not repeat here.

Next, we start on the HeapSort algorithm. Since most of the students already know what is heap, we will focus on analysis part. See Chapter 6 if you forget what is heap, covered in data structure course. There are several key facts about heap. The most important one is, the height $H_t$ of the heap is $O(\log n)$. To see this, we note that at level $i$, there are no more than $2^i$ nodes. Here, we say the root is at level 0 (where the root has height $H_0$). Also note the heap forms a complete binary tree. Thus, we have the number of nodes $n \leq \sum_{i=0}^{H_t} 2^i = 2^{H_t+1} - 1$. Also, if we omit the last term in the above summation, we have: $n \geq 1 + \sum_{i=0}^{H_t-1} 2^i = 2^{H_t}$. Thus, $\log n - 1 \leq H_t \leq \log n$.

Based on this fact, the algorithm Heapify(A, i) will run $O(\log n)$ time since it only processes up to $\log n$ nodes when pushing the small value downwards, each time takes constant time. We conclude with the algorithm for building the heap from an unordered array A by repetitively calling Heapify(A, i).