Math 261C: Randomized Algorithms

Lecture topic: k-th element/median summary & WalkSAT

Lecturer: Sam Buss
Scribe notes by: James Aisenberg
Date: April 4, 2014

1. k-th element/median summary

Number of comparisons

(1) QuickSelect
  - Number of comparisons: $2(1 + \ln 2)n + o(n) \approx 3.386n$
  - number of random bits: $O(\log^2 n)$ with high probability

(2) FR-Select
  - Number of comparisons: $n + \min\{k, n - k\} + o(n) \leq \frac{3}{2}n$
  - number of random bits: $O(\sqrt{n}\log n)$ with high probability

(3) D-Select – "D" = deterministic
  - Number of comparisons: $\approx 24n$ (for the algorithm sketched in class), $\approx 5.43n$ [SPP '76], best known $\approx 2.94n$ [D-Z '99]
  - number of random bits: 0

2. Satisfiability and WalkSAT

SAT is the canonical NP-complete problem. Some conventions and terminology for SAT:

- *Variables* $x_0, x_1 \ldots$ that take on true/false values.
- A *literal* is either a variable x, or its negation, $\neg x$ or $\bar{x}$.
- A *clause* is a finite set of literals. It denotes the *disjuntion* or, "or", of the literals. For example, $\{x_3, \bar{x_7}, x_9\}$ means $x_3 \vee (\neg x_7) \vee x_9$.
- A finite set of clauses, Γ denotes the conjunction or, "and", of the the clauses.
- The *satisfiability (SAT) problem* is: given Γ a set of clauses, is there a truth assignment $\tau : \{x_0, \ldots\} \to \{\top, \bot\}$ to the variables that makes every clause $C \in \Gamma$ true?

- k-SAT is the satisfiability problem restricted to clauses of size less than or equal to k.

Fact: SAT is NP-complete. If SAT is in P (polynomial time), then so is every NP problem.

The naive algorithm for SAT: try all combinations. If there are n variables, $x_0, \ldots, x_{n-1}$, then there are 2^n many τ's to try. The runtime is $\approx |\Gamma| \cdot 2^n$.

The naive randomized algorithm for SAT is as follows.

loop
  Choose τ at random
  if τ satisfies Γ **then**
    Output τ
  end if
end loop

Observe that if there are s satisfying assignments τ, then each loop can succeed with probability $\frac{s}{2^n} = p$ So then the expected number of iterations is $1/p$. Thus, the expected runtime is $\leq |\Gamma| 2^n$ (i.e., no better than the deterministic algorithm).

WalkSAT improves on this. It is due to Schöning [1], and is similar to "GSAT" of Kautz *et al.* [2]. It will have runtime $|\Gamma| \left(2 \cdot \frac{k-1}{k}\right)^n$ for Γ, an instance of k-SAT.

First we will give an algorithm for one iteration of WalkSAT, it takes as input Γ an instance of k-SAT in n variables, and m, a parameter. It either outputs a satisfying assignment, or no output.

procedure WalkSatIteration(Γ,m)
  Choose a truth assignment, τ at random
  loop m times
    if $\tau \vDash \Gamma$ (τ satisfies Γ) **then**
      Output τ and halt.
    else
      Pick any clause $C \in \Gamma$ such that τ does not satisfy C
      In C, pick some literal x at random. $\triangleright$ So in particular $\tau(x) = \bot$.
      Modify τ by flipping its value on that literal
    end if
  end loop
end procedure

To analyse this algorithm, we want a lower bound on the probability p that WalkSAT finds a satisfying assignment (assuming Γ is satisfiable). We'll show

$$p \geq \left(\frac{1}{2}\frac{k}{k-1}\right)^n \frac{1}{|\Gamma|^{O(1)}}$$

Let π be some fixed satisfying assignment. Let $d(\tau,\pi)$ be the "Hamming distance." In other words, the number of x_i's where $\tau(x_i) \neq \pi(x_i)$.

Take some C, a clause falsified by τ. C has $k' \leq k$ many literals. On the other hand $\pi \vDash C$. We choose x at random in C. With probability l/k', $\pi(x) = \top$ (of course, $\tau(x) = \bot$). Let τ' be the "new" τ, so $\tau'(x) \neq \tau(x)$. So then $d(\tau,\pi) = d(\tau,\pi) - 1$ with probability $l/k' \geq 1/k$. and $d(\tau',\pi) = d(\tau,\pi) + 1$ with probability $\frac{k'-l}{k'} \leq \frac{k-1}{k}$.

To complete the proof: We describe a Markov process on $n+1$ states: $0,\ldots,n$. If in state $j < n-1$, the probability of going to $j+1$ is $\frac{k-1}{k}$ and the probability of going to $j-1$ is $\frac{1}{k}$. If in state n, then the probability of transitioning to state $n-1$ is 1.

Success is defined by reaching state state 0. We'll bound p by bounding q_j, which is the probability of succeeding (reaching state 0) in $\leq 3n$ many steps when starting in state j.

The overall WalkSAT algorithm is as follows.

```
procedure WalkSat(Γ,n)
    loop                                  ▷ Sufficiently many times, about 1/p many times
        WalkSatIteration(Γ,3n)
        if WalkSatIteration returns a satisfying assignment then
            Return that satisfying assignment
        end if
    end loop
end procedure
```

We will analyze the the success probability of the WalkSAT algorithm in the next lecture.

References

[1] Schöning, *A probabilistic algorithm for k-SAT and constraint satisfaction problems*, 40th Symp. on Foundations of Computer Science (FOCS), 1999, pp. 410–414.

[2] Kautz and B. Selman *Pushing the envelope: planning, propositional logic, and stochastic search*, Proceedings of the Thirteenth National Conference on Artificial Intelligence, 1996, pp. 1194-1201.