Lecture 6

Lecturer: Mark Braverman *Scribe: Yonatan Naamad**

1 A lower bound for perfect hash families.

In the previous lecture, we saw that the cardinality t of a k-perfect hash family $\mathcal{H} = \{[N] \to [b]\}$ must satisfy the inequality

$$t \geq \frac{\log N}{\log b} \tag{1}$$

Heuristically, this makes sense: as we increase b, we're only relaxing the problem by increasing the number of values to which we can map the keys, so t can only decrease. Conversely, increasing N only increases the difficulty of finding an appropriate family of hash functions, so t must increase accordingly. What equation 1 doesn't capture, however, is that an increase in k should also result in an increase in t, with a particularly notable increase as k approaches b (the problem being infeasible for $k > b$). This relationship is captured in the following theorem

Theorem 1. *Any k-perfect hash family $\mathcal{H} = \{[N] \to [b]\}$ of cardinality t must satisfy*

$$t \geq \frac{b^{k-1}}{b(b-1)\cdots(b-k+2)} \cdot \frac{\log(N-k+2)}{\log(b-k+2)} \tag{2}$$

Proof

This theorem was first proven by Fredman and Komlós in '84. This information theoretic proof is by Körner, from '86.

For simplification, assume $b|N$. Let G denote the following graph:

- Vertices of G: $\{(D, x) : D \subseteq [N], |D| = k-2, x \in [N] - D\}$
- Edges of G: $\{\{(D, x_1), (D, x_2)\} : x_1 \neq x_2\}$

From the definition, we see that G has one connected component for each of the $\binom{N}{k-2}$ possible values of D, with each such component being a clique of size $N - k + 2$. For each $h \in \mathcal{H}$, define the following subgraph $G_h \subset G$

- Vertices of G_h: $\{v : v \text{ is a vertex of } G\}$
- Edges of G_h: $\{\{(D, x_1), (D, x_2)\} : x_1 \neq x_2, h \text{ is injective on } D \cup \{x_1, x_2\}\}$

Each edge corresponds to k points, and the subgraph G_h contains the collection of edges on which h is injective on all k of their points. This, in conjunction with the fact that $\mathcal{H}$ is k-perfect, implies that

$$G = \bigcup_{h \in \mathcal{H}} G_h$$

Because each component of G is a clique of size $N - k + 2$, the entropy of each connected component of G is $\log(N - k + 2)$, so the entropy of G itself is also $\log(N - k + 2)$.

*Based on lecture notes by Anup Rao and Lukas Svec

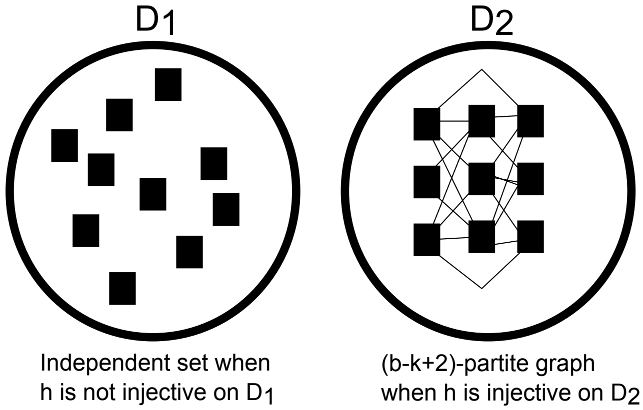

Figure 1: Types of components forming the structure of G_h

To bound t, we will find an L such that $H(G_h) \leq L$, which in turn implies that $t \geq \frac{\log(N-k+2)}{L}$ by subadditivity of graph entropy.

Consider the various connected components of G_h for any fixed h. Each such component corresponds to a subset $D \subseteq N$ of size $k-2$. If h is *not* injective on D, then the connected component is empty. If h *is* injective, then for each $i \notin h(D)$, define $A_i = \{(D, x) : h(x) = i\}$. The component corresponding to D only has edges going between A_i and A_j when $i \neq j$, so it is a $(b-k+2)$-partite graph. Thus,

$$H(\text{each connected component}) \leq \log(b-k+2)$$

so $H(G_h) \leq \log(b-k+2)$, implying that

$$t \geq \tfrac{\log(N-k+2)}{\log(b-k+2)}. \tag{3}$$

This is already a much better bound than that given by equation 1, but it can be further improved by more closely examining the structure of each G_h, shown in figure 1. Specifically, we will exploit the fact that G_h has a large number of isolated vertices to improve our upper bound on its entropy, and thereby tighten our lower bound on t.

Ideally, we'd want to figure out how many isolated vertices (D, x) are there in G_h. Note that (D, x) is isolated iff h is not injective on $D \cup \{x\}$. Since each set S of size $k-1$ such that h is not injective on S gives rise to $k-1$ isolated vertices, the fraction of isolated vertices in G is equal to the probability of having h not be injective on S for some randomly chosen S of size $k-1$.

By simple combinatorics, the total number of vertices in G_h is given by

$$\binom{N}{k-2} \cdot (N-k+1) = \binom{N}{k-1} \cdot (k-1)$$

To calculate the probability that h is injective on S, we use the following fact

Claim 2. $\Pr_{|S|=k-1}[h \textit{ is injective on } S]$ *is maximized when h partitions $[N]$ evenly.*

Sketch of proof:
The given statement is equivalent to stating that the probability is maximized when

$$|h^{-1}(1)| = |h^{-1}(2)| = \cdots = |h^{-1}(b)|.$$

Assume to the contrary that (without loss of generality) there is a higher probability of h being injective S for some h with $|h^{-1}(1)| > |h^{-1}(2)|$. Let x be an arbitrary element in $h^{-1}(1)$, and let's see what happens to

$\Pr_{|S|=k-1}[h \text{ is injective on } S]$ if we were to change $h(x)$ from 1 to 2. We can write

$\Pr[h \text{ is injective on } S] = \Pr[x \in S] \cdot \Pr[h \text{ is injective on } S | \, x \in S] + \Pr[x \notin S] \cdot \Pr[h \text{ is injective on } S | x \notin S]$

As the first term in the summation can only increase with the change and the second term is independent of changes in $h(x)$, the probability of h being injective was increased by this change. Thus, our original h could not have been the probability maximizing partition, so we conclude that claim 2 must hold. □

Thus, the probability that h is indeed injective is equal to the probability that $k-1$ elements, each placed independently and uniformly at random in to one of b buckets, all fall in different buckets. This probability is given by

$$p = \Pr[h \text{ is injective on } S] = 1 \cdot \frac{b-1}{b} \cdot \frac{b-2}{b} \cdots \frac{b-k+2}{b} \tag{4}$$

Thus, each G_h consists of two parts

1. A disjoint union of $(b-k+2)$-partite graphs, each of which has at most $\log(b-k+2)$ entropy.
2. p isolated vertices, each of which has 0 entropy.

Therefore, the entropy of G_h, which is the weighted average of the entropy of its components, is given by

$$\begin{aligned} H(G_h) &\leq \log(b-k+2) \cdot \Pr[\text{uniformly chosen vertex is not isolated}] \\ &\leq \log(b-k+2) \cdot \frac{b(b-1)\cdots(b-k+2)}{b^{k-1}} \end{aligned}$$

The originally sought inequality follows from $t \geq \dfrac{\log(N-k+2)}{H(G_h)}$. ■

2 Circuit/Formula Complexity

2.1 Monotone boolean formulas and functions

Definition 3 (Boolean formula). *A boolean formula on inputs $x_1, \cdots x_n$ is a rooted tree with each leaf being an element of $\{x_1, \cdots, x_n, 0, 1\}$ and each internal node corresponding to one of the boolean functions `AND`, `OR`, or `NOT`.*

Example 4 (Boolean formula for `XOR`). *The boolean formula for `XOR` is given by figure 2.*

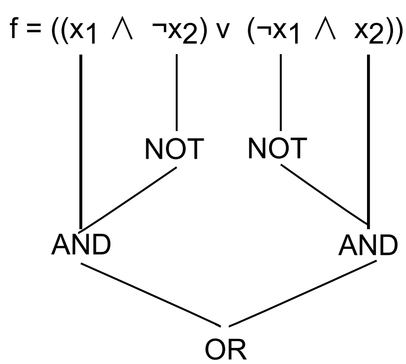

Figure 2: Sample boolean circuit for computing `XOR`

We say that the *size* of a boolean formula is the total number of vertices (including leaves) in the corresponding tree.

Definition 5 (Size of a function)**.** *The size of a function* $f : \{0,1\}^n \to \{0,1\}$ *is the size of the smallest boolean formula computing* f*.*

A simple counting argument shows that most functions have large sizes, but in general it is very difficult to prove any explicit lower bounds.

Definition 6 (Monotone formula)**.** *A formula is called monotone if it only uses* `AND` *and* `OR` *gates.*

Alternatively, a formula f is monotone if for all $x_1, \cdots, x_n$ and for all $i \in \{1, \cdots, n\}$

$$f(x_1, \cdots, x_{i-1}, 0, x_{i+1}, \cdots, x_n) \leq f(x_1, \cdots, x_{i-1}, 1, x_{i+1}, \cdots, x_n)$$

For example `XOR` is *not* monotone, because there are instances in which flipping a bit from 0 to 1 changes the truth value of the formula from 1 to 0.

To allow us to think of boolean formulas in terms of operations on sets, identify $S \subseteq \{1, \cdots, n\}$ with binary vectors such that $x_i = 1 \leftrightarrow i \in S$.

Claim 7. f *is monotone iff* $f(S) \leq f(T)$ *whenever* $S \subseteq T$*.*

Proof This follows from the fact that

$$f(x_1, \cdots, x_n) = \bigvee_{\{S|f(S)=1\}} \left(\bigwedge_{i \in S} x_i \right).$$

This can be shown as follows: let $T = \{i | x_i = 1\}$. If $f(T) = 1$, then $\bigwedge_{i \in T} x_i = 1$ because T is one of the clauses in the DNF formula. However, if the formula returns a 1, then there exists a set $S \subseteq T$ such that $f(S) = 1$, which by monotonicity implies $f(T) = 1$. ■

For any monotone function f, define $\text{size}_m(f)$ to be the smallest monotone formula computing f. Clearly, $\text{size}_m(f) \geq \text{size}(f)$, as the latter is simply a relaxation of the former.

2.2 Threshold function

Definition 8 (Threshold functions)**.** *The threshold function is defined as* $\text{Th}_k^n(S) = \begin{cases} 1 & \text{if } |S| \geq k, \\ 0 & \text{otherwise.} \end{cases}$

Example 9 (Simple examples of threshold functions)**.**

$$\text{Th}_1^n(S) = x_1 \vee \cdots \vee x_n \qquad \text{size}(\text{Th}_1^n) = 2n - 1$$
$$\text{Th}_n^n(S) = x_1 \wedge \cdots \wedge x_n \qquad \text{size}(\text{Th}_n^n) = 2n - 1$$

It turns out that the threshold function of largest size is the majority function $\text{Th}_{n/2}^n$, which is of size $\mathcal{O}(n^{5.3})$ (Valiant, '84). Instead of computing this, however, we begin by trying to calculate the size of Th_2^n.

The most intuitive formula for computing this function is simply $\bigvee_{i \neq j}(x_i \wedge x_j)$, which is of size $\mathcal{O}(n^2)$. However, we can employ a divide-and-conquer approach to reduce the size to $\mathcal{O}(n \log n)$. This can be done as follows:

1. Divide the input X into two parts Y,Z each of size $n/2$.
2. Recursively compute $\text{Th}_2^n(Y, Z) = \text{Th}_2^n(Y) \vee \text{Th}_2^n(Z) \vee (\text{Th}_1^n(Y) \wedge \text{Th}_1^n(Z))$

Intuitively, the last formula states that at least two bits are set exactly when either Y contains at least 2 set bits, Z contains at least 2 set bits, or each of Y and Z contain at least 1 set bit. Because the last term in the above formula is of size $\mathcal{O}(n)$, the size of this formula, $\text{size}^*_m(\text{Th}^n_2)$ satisfies the recurrence relation

$$\text{size}^*_m(\text{Th}^n_2) = 2 \cdot \text{size}^*_m(\text{Th}^{n/2}_2) + \mathcal{O}(n)$$

which, by the same analysis as that of `mergesort` gives that $\text{size}^*_m(\text{Th}^n_2) = \mathcal{O}(n \log n)$, giving us the sought upper bound on $\text{size}_m(\text{Th}^n_2)$. As shown by Krichevski in '64, $\text{size}_m(\text{Th}^n_2) \geq 2\lceil n \lg n \rceil - 1$, which was later shown to hold at equality Newman, Ragde, and Widgerson in '90, to be presented next class.