Today

- bipartite matching proof
- intro hashing
- universal hashing

Takeaways:

- often make decisions with incomplete info.
  - competitive analysis gives us a way to bound cost of not knowing future
  - competitive ratio is a worst-case notion but online algorithms with
    best possible competitive ratio often work much better in practice

- "ascending auction" alg for matching (or selling goods to buyers)

Today we will prove that this very natural, simple alg probably has very nice properties.
**Maximum Weight Matching**

bipartite graphs

matching: subset of edges with no common endpoints

match is perfect if all vertices are incident to an edge in matching

Algorithmic problem:

find a maximum weight matching in a weighted $m \times n$ bipartite graph. 

\[\begin{array}{ccc}
1 & 2 & 3 \\
\hline 
4 & 5 & 6 \\
\end{array}\]

assume weights $w_{ij}$ are integers

Fix bid increment $\delta = \frac{1}{m+n}$

Maintain price vector $(p_1, \ldots, p_n)$ 

Initially all prices $p_j = 0$ and matching is empty $M(i) = \emptyset$

As long as some bidder is not matched

pick unmatched bidder $i$ 

consider $D(i) = \{ j \mid v_{ij} \geq p_j \}$ 

Pick some $j \in D(i)$

If $j$ unmatched, say $M(i) = j$

else, say $M(i) = \emptyset$, $M(j) = \emptyset$, $M(i) = j$

increase $p_j$ by $\delta$.
Theorem: Suppose run this alg on \( M \) complete bipartite graph with integer weights. Then the alg terminates with a perfect max weight matching \( M \) and the prices are almost "envy-free" i.e.

\[
M(i) = j \implies v_{ij} - p_i > v_{ik} - p_k > -5w_k
\]

Proof:

Observations:
- From the moment an item is matched, it stays matched until item \( j \) is matched, \( p_j = 0 \)
- if bidder \( i \) is unmatched, then \( D_i(G) \neq \emptyset \)
- every step \( \Delta(\Sigma p_i) = 5 \)
- \( p_k \leq \max_v V_k + 5 \)
- \( \implies \) all bidders are matched at end.

When we terminate property (a) holds.

Suppose that \( M^{\textrm{opt}} \) is an optimal

\[ \sum_i v_i \cdot m_i(i) \geq \sum_i v_i \cdot m_i(i) - 5 \cong \sum_i v_i \cdot m_i(i) - 5 \]

Now we will show

\[ \sum_i v_i \cdot m_i(i) - p_i(i) \geq \sum_i (v_i \cdot m_i(i) - 5) \]

\[ \sum_i v_i \cdot m_i(i) \geq \sum_i (v_i \cdot m_i(i) - 5) \]

\[ \delta = \frac{1}{n} \sum_i v_i \cdot m_i(i) - \frac{n}{n} \]

\[ \delta = \frac{1}{n} v_i \cdot m_i(i) - \frac{n}{n} \]

\[ \delta = \frac{1}{n} v_i \cdot m_i(i) - \frac{n}{n} \]

unhappy a bidder had for an item

\[ v_{ij} - p_i \geq v_{ij} - p_j \]

OPT \(
\)
\[ p_i = c + \delta \]
\[ v_{ij} \]
\[ \max_{i,j} v_{ij} + 5 \]
\[ c \]

\[ n = \# \text{bidders} = \# \text{dems} \]
\[ \max(\# \text{bidders}, \# \text{dems}) \]

\[ c = 1 \]
\[ s = \frac{1}{n} \]

\[ \text{running \ } O(n^2) \]

\[ n = 2 \]
\[ \delta = \frac{1}{3} \]
Hashing

One of the most important ways to implement:
- dictionary data structure
- load balancing
- numerous applications in algorithms, complexity, and crypto

Dictionary

U = universe of possible keys
e.g. \{0, ..., 2\^n-1\}

Operations:
- Insert(k) - add key k to set
- Find(k) - is k in set?
- Delete(k) - delete k from set

Want to store set S of keys from U

ideally like to use space \( O(|S|) \)

\(|S| = n \)

\[ h(k) \]

\[ m \]

collisions: when two keys hash to same location

Time for Find(k) = # keys that hash to \( h(k) \)

can be linear in worst case

Desiderata

- # sets in each bucket is small, ideally \( O(1) \)
- \( m = O(|S|) \)
- efficient to store & compute \( h(\cdot) \)

Start by assuming that \( h \) is completely random.

\[ \mathcal{N} = \text{set of all mappings} \]

\[ |\mathcal{N}| = m^n \]

\[ \Pr(\text{any particular mapping}) = \frac{1}{m^n} \]

Find(x)

T: # els stored at location \( h(x) \)

if \( x = y \):
Define \( H \) a family of hash functions (each \( h \in H \) \( h: U \rightarrow \{0,1\}^m \)) such that

\[
\Pr(h(x) = h(y)) \leq \frac{1}{m}
\]

for a random choice of \( h \in H \).

If we have universal family of hash functions, we pick \( h \) UK

\[ |U| = 2^u \quad m = 2^k \]

\[ u = 12.8 \quad k = 10 \]

\( H = \{ h_A(x) = A x \mod 2 \mid A \in \{0,1\}^{k \times u} \} \)
\[ h_A = A \times \text{mod} 2 \]

pick one \( A \) at random

\[ h_{A'} \]

Thm

If \( h \) is selected at random from this family

(A is random \( k \times n \) matrix
glove)

then \( \forall x \neq y \in U \)

\[ Pr(h_A(x) = h_A(y)) = \frac{1}{m} \]

\[ \mathcal{X} = \{ h_{A(x)} = A \times \text{mod} 2 \mid A \in \{0, 1\}^{k \times n} \} \]

\( h_A : U \rightarrow [m] \)

\[ |U| = 2^n \quad m = 2^k \]

Claim: universal

Suppose \( x \neq y \)

\[ Pr(Ax = Ay) = Pr(A(x-y) \text{mod} 2 = 0) = \frac{1}{2^k} \frac{1}{m} \]

\[ z \neq 0 \quad \text{(because } x \neq y) \]

\[ z = (z_1, \ldots, z_n) \]
\[
\begin{align*}
\text{at least one } z_i = 1 \implies z = i \\
\Pr(\ c_i+a_i=0 \mod 2) &= \frac{1}{2^k} \\
\Pr(\ (c_i+a_i) \mod 2 = 0) &= \frac{1}{2} \\
\end{align*}
\]