Last time

- online algs
- bipartite matching auction alg
- mini-probability review

Today

- bipartite matching - proof
- intro hashing
- universal hashing

Takeaways:

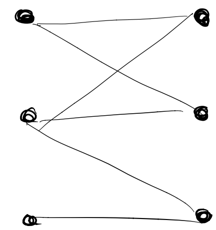
- often make decisions with incomplete info.
  - competitive analysis gives us a way to bound cost of not knowing future
  - competitive ratio is a worst-case notion but online algorithms with best possible competitive ratio often work much better in practice

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

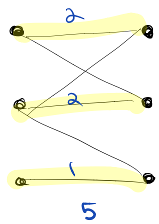
  today we will prove that this very natural, simple alg provably has very nice properties.

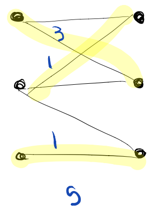
Maximum Weight Matching

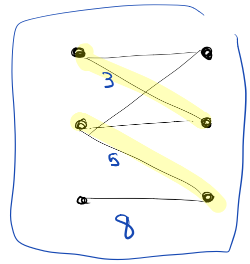
bipartite graphs



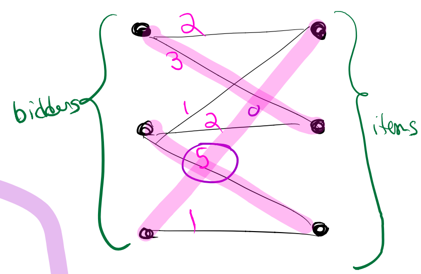
matching: subset of edges with no common endpts







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



Algorithmic problem:

find a maximum weight matching in a weighted $n \times n$ bipartite graph. [if $\nexists$ edge (i,j), set $v_{ij}=0$]

v_{ij} – value bidder i has for item j

assume weights (v_{ij}'s) are integers

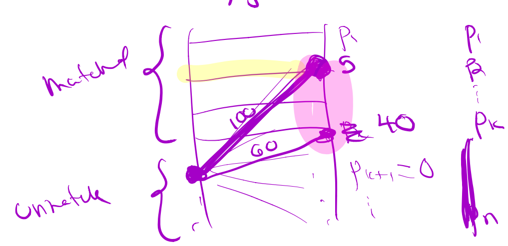
ascending auction algorithm

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

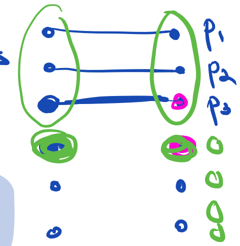
Maintain price vector $(p_1, \dots, p_n)$ — p_j is the price of item j

Initially all prices $= 0$ and matching is empty. $M(i)$ [matching bidder i] $M(i) = \phi$.

As long as some bidder is not matched
- pick unmatched bidder i
- consider $D(i) = \left\{ \text{items } j \;\middle|\; v_{ij} - p_j \geq v_{ik} - p_k \;\; \forall k \neq j, \;\; v_{ij} \geq p_j \right\}$
- Pick some $j \in D(i)$
- If j unmatched, then $M(i) := j$
- else, say $M(\ell) = j$,
  - $M(\ell) := \phi$
  - $M(i) := j$
- increase p_j by δ.



Theorem Suppose run this alg on $n \times n$ 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.



$$\boxed{M(i) = j \implies v_{ij} - p_j \geq v_{ik} - p_k - \delta \quad \forall k} \quad (*)$$

Proof

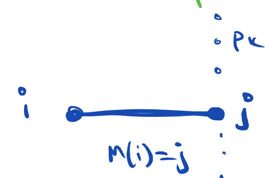
Observations:

- from the moment an item is matched, it stays matched
- until item j is matched, $p_j = 0$
- if bidder i is unmatched, then $D_i(\vec{p}) \neq \phi$

$$D(i) = \left\{ \text{items } j \;\middle|\; \overline{v_{ij} - p_j} \geq \overline{v_{ik} - p_k} \;\; \forall k \neq j \quad v_{ij} \geq p_j \right\}$$

- every step $\Delta(\sum p_j) = \delta$
- $p_k \leq \max_i v_{ik} + \delta$

$\implies$ all bidders are matched at end.



When we terminate property $(*)$ holds

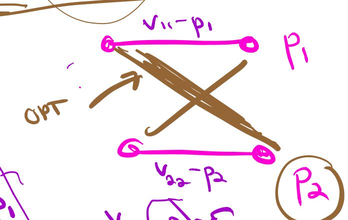
when i was matched to j for final time $j \in D_i(p)$

Suppose that M^* is an optimal max wt matching.

$$\sum_i v_{iM^*(i)} \geq \sum_i v_{iM'(i)} \quad \forall \text{ matchings } M'$$

now we will show

$$\sum_{i=1}^{n} (v_{iM(i)} - p_{M(i)}) \geq \sum_{i=1}^{n} (v_{iM^*(i)} - p_{M^*(i)} - \delta) \quad (*)$$



$$\sum_{i=1}^{n} v_{iM(i)} \geq \sum_{i=1}^{n} (v_{iM^*(i)} - \delta)$$

$$= \sum_{i=1}^{n} v_{iM^*(i)} - n\delta$$

$\delta = \frac{1}{n+1}$

$$= \sum_{i=1}^{n} v_{iM^*(i)} - \frac{n}{n+1}$$

$$\begin{aligned} &v_{11} - p_1 \\ &+ v_{22} - p_2 \end{aligned} \geq \begin{aligned} &v_{12} - p_2 - \delta \\ &+ v_{21} - p_1 - \delta \end{aligned}$$

$$v_{11} + v_{22} - (p_1 + p_2) \geq v_{12} + v_{21}$$

utility a bidder has for an item j: $v_{ij} - p_j$

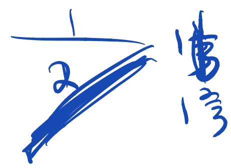
max price $\le \underbrace{\max_{i,j} v_{ij}}_{C} + \delta$

$p_j = C + \delta$

$n\left(\frac{C+\delta}{\delta}\right)$

$C = 1$ $\quad \delta = \frac{1}{n+1}$

running time $O(n^2)$



$n =$ #bidders $=$ #items

max(#bidders, #items)

$n = 2$

$\delta = \frac{1}{3}$

Hashing

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

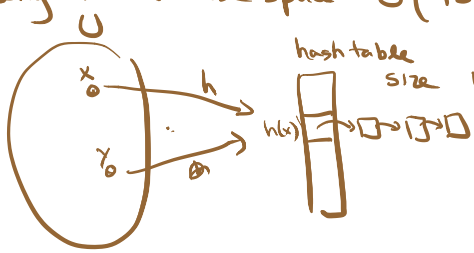
Dictionary

U - universe of possible keys
e.g. $\{0, \dots, 2^{32}-1\}$

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 $O(|S|)$ $\quad |S| = n$



collisions: when elts hash to same location.

Time for Find(k) - # keys that hash to $h(k)$
can be linear in worst case

Desiderata
- # elts 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.

Ω : set of all mappings from $[n] \to [m]$

$|\Omega| = m^n$

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

$$\Pr(h(x_i) = k_i \ , h(x_j) = k_j)$$
$$= \Pr(h(x_i) = k_i) \Pr(h(x_j) = k_j)$$
$$= \frac{1}{m} \cdot \frac{1}{m}$$

$$\Pr(h(x_i) = k) = \frac{1}{m}$$

$$\Pr(\ ) = \frac{\# \text{mappings } x_i \to k}{\# \text{mappings}} = \frac{m^{n-1}}{m^n} = \frac{1}{m}$$

Find(x)

T: # elts stored at location $h(x)$ $\quad$ if $x = x_i$

$$T = \sum_{j=1}^{n} X_j \qquad X_j = \begin{cases} 1 & x_j \text{ maps to same location as } x \\ 0 & \text{o.w.} \end{cases}$$

$$\boxed{\begin{aligned} E(T) &\le 1 + \sum_{j=1}^{n} E(X_j) \\ &\le 1 + \frac{n}{m} \end{aligned}} \qquad \boxed{\begin{aligned} E(X_j) &= \Pr(h(x_j) = h(x)) \\ &= \frac{1}{m} \end{aligned}}$$

$m = \Omega(n)$

$$\boxed{E(T) = O(1)} \qquad E(T) = 1 + \sum_{x_j \ne x} E(X_j)$$

x_1	$h(x_1)$
x_2	$h(x_2)$
$\vdots$	
x_n	$h(x_n)$

$x \bmod m$

Carter & Wegman $\mathcal{H}$

Defn $\underline{\mathcal{H}}$ a family of hash fns (each $h \in \mathcal{H}$ $h: U \to [m]$) is <u>universal</u> if $\forall\, x \ne y \in U$

$$\Pr(h(x) = h(y)) \le \frac{1}{m}$$

over a random choice of $h \in \mathcal{H}$

If we have universal family of hash fns & pick $h \in \mathcal{H}$ u.a.r. use h to hash set S of n elements, then

$\forall x$ $E(\#\text{elts that collide with } x) \le 1 + \frac{n}{m}$

$$|U| = 2^u \qquad m = 2^k$$

$u = 128$ $\qquad$ $k = 10$

$$\boxed{\mathcal{H} = \{ h_A(x) = A x \bmod 2 \mid \underline{A} \in \{0,1\}^{k \times u} \}}$$

$x \in U$ $\qquad$ $\boxed{2^{ku}}$

$$\begin{array}{c} \\ k \end{array} \overbrace{\begin{pmatrix} 0 & 1 & 1 & \cdots & 0 \\ 0 & 1 & \cdots & & 0 \\ \cdots & & & & \end{pmatrix}}^{u} \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

A $\qquad$ $\leftarrow x$ $\qquad$ $x \in U$

$h_A = Ax \bmod 2$

$\boxed{\text{pick one } A \text{ at random}}$

h_A

$\begin{pmatrix} 0 \\ 0 \end{pmatrix}_u$

$(0\ 1\ 1)\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = 0 \bmod 2$

1,280 bits at random

$x_1, x_2, \dots, x_n \in$ hashtable

$Ax_1 \bmod 2,\ Ax_2 \bmod 2 \dots$

Thm If h is selected at random from this family (A is random $k \times u$ matrix g bits)

then $\forall x \neq y \in U$ $\boxed{\Pr(h_A(x) = h_A(y)) = \frac{1}{m}}$

2^k

$\mathcal{H}_1 = \{h_{11}, h_{12}, h_{13}, \quad h_{1r}\}$

$U \to [m]$

$\{h_{A_1}, h_{A_2} \quad \dots \quad h_{A_{2^{ku}}}\}$

$\mathcal{H} = \{ h_A(x) = Ax \bmod 2 \mid A \in \{0,1\}^{k \times u}$

$h_A : U \longrightarrow [m]$

$|U| = 2^u \qquad m = 2^k$

<u>Claim: universal</u>

Suppose $\boxed{x \neq y}$

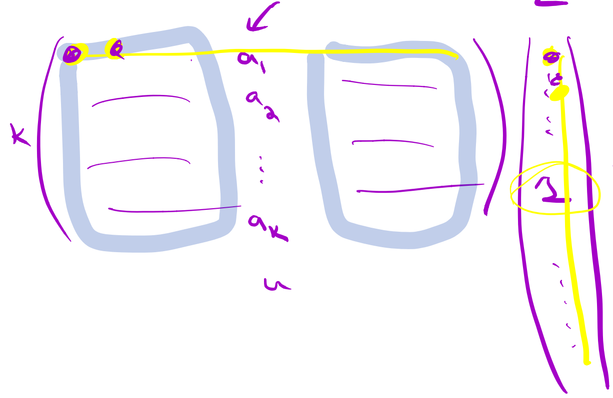
$\Pr(Ax = Ay) = \Pr(A\underbrace{(x-y)}_{z} \bmod 2 = 0) = \frac{1}{2^k} = \frac{1}{m}$

random choice of matrix A

$z \neq 0$ (because $x \neq y$

$z = (z_1, \dots, z_u)$

$\exists$ at least one i s.t. $z_i = 1$



$$= \begin{pmatrix} c_1 + a_1 \\ c_2 + a_2 \\ \vdots \\ c_k + a_k \end{pmatrix} \bmod 2$$

$c_1 = 1$

$c_1 = 0$

$$\Pr\left(c_i + a_i = 0 \bmod 2 \ \ \forall\ 1 \le i \le k \right) = \frac{1}{2^k}$$

$a_i = c_i \bmod 2$

iff $\quad a_i + c_i = 0 \bmod 2$

$$\Pr\left((\overset{1}{c_i} + \overset{1}{a_i}) \bmod 2 = 0 \right) = \frac{1}{2}$$

over choice of a_i

$\blacksquare$