Recall: online algorithm - handles sequence of requests as they come.

Alg. $A$ is **c-competitive** if

$$A(\sigma) \leq c \cdot \text{OPT}(\sigma) + b$$

for min problem

and if $b$ is constant,

$$A(\sigma) \geq c \cdot \text{OPT} - b$$

for max problem.

**k-Server Problem** - $k$ servers to service requests in metric space of points $P_1 \ldots P_n$

- Request for $P_i$
  - if a server is at $P_i$: fine
  - else move a server from its location, say $P_j$, to $P_i$ at cost $d(P_j, P_i)$ (distance).

Goal: serve requests in given order and minimize total distance.

The offline $k$-server problem can be solved in poly. time via dynamic programming.

Paging is special case - the "points" are the pages in slow memory, "serving" a request = putting the page in the cache, distances are all 1.

**Local Greedy Algorithm**

- To meet next request, say at $P$, move the closest server, i.e. move server from $q$ to $P$, to minimize $d(P,q)$

Claim: This is not $c$-competitive for any $c$. 

Pf. Consider point $p_1, p_2, q$ on a line.

$\circ o \circ \circ \circ$

does server

$P_1, P_2$  

2 servers initially at $P_2$ and at $q$.

request sequence: $P_1, P_2, P_1, P_2$...

Greedy moves one server between $P_1$ and $P_2$.

cost = length of sequence

OPT moves server from $q$ to $P_1$. — cost is constant.

Ex. What does LRU do for $k$-server? Find a bad example: $d(q, P_1)$

Conjecture (open since 1988): There is a $k$-competitive alg. for $k$-server problem.

Two cases with $k$-competitive alg.: 1. paging (LRU)

2. the following.

Double-Coverage Alg.

Case 1: if request is to right [left] of all servers, move closest server.

Case 2: if request is between 2 servers, move both until one reaches the request. (the other will stop at a non-request point.)

(if multiple servers at same point, break ties arbitrarily.)

Then, this is $k$-competitive.

E.g. on above bad example:

request $\Rightarrow P_1, P_2, P_2$  

$\circ o \circ \circ \circ$

Note that $q$ starts moving left.

Eventually $q$ will reach $P_2$ and then all requests are free.
Pf of theorem. We need to compare ALG to OPT will use an amortized analysis with potential $\Phi$. Think of ALG having $k$ servers and OPT having $k$ servers. Think of $OPT$ moves, then $ALG$ moves (at the $i$th request) $\Phi$ will depend on the difference. Properties we will ensure:

1. $\Phi_i \geq 0$
2. If move of $OPT$ costs $s_i$, then incr. of $\Phi$ is $\leq k \cdot s_i$
3. If moves of $ALG$ cost $t_i$, then incr. of $\Phi$ is $\leq -t_i$ \( \text{(i.e. $\Phi$ decreases by $\geq t_i$)} \)

Then $\Phi_{i+1} - \Phi_i \leq k \cdot s_i - t_i$

Taking sum:

$$\Phi_f - \Phi_0 \leq k \cdot \sum s_i - \sum t_i$$

$\sum s_i$ \( \text{OPT cost of ALG} \)

$$ALG \leq k \cdot OPT + \Phi_0 - \Phi_f \leq k \cdot OPT + \Phi_0$$

How do we define $\Phi$?

$$\Phi = k \cdot M + D$$

$D$ = sum of all $\binom{k}{2}$ distances between pairs of Alg's servers.

$M$ = min matching between Alg's servers and Opt's servers.

\[ M = d \]

**E.g. 1 server each**

\[ M = d \]

**E.g. 2 servers each**

The red matching has weight $d_1 + d_3$

The outer matching has weight $(d_1 + d_2 + d_3) + d_2$

So $M = d_1 + d_3$
Proving the properties:

1. \( \phi_i \geq 0 \) since \( M, D \geq 0 \)

2. Suppose \( \text{OPT} \) moves a server distance \( s_i \)
   Then \( D \) unchanged, and \( M \) increases by \( \leq s_i (\leq k \cdot s_i) \)
   (we need \( k \cdot s_i \) later)

3. Suppose \( \text{ALG} \) moves cost \( t_i \)
   
   2 cases:

   Case 1: \( \text{ALG} \) moved rightmost (or leftmost) server
   
   \[ \begin{array}{c}
   s \quad \rightarrow \quad x \quad \quad \text{OPT already put a server there, so} \\
   t_i \end{array} \]
   
   \( D \) goes up by \( (k-1) t_i \) since server \( s \) moves away
   
   \( \phi \) decreases by \( \geq k t_i - (k-1) t_i = t_i \)

   Case 2: \( \text{ALG} \) moved two servers cost \( t_i = 2d \)
   
   \[ \begin{array}{c}
   s_1 \quad \rightarrow \quad x \quad \quad \text{one match decreases} \\
   d \quad \quad \quad \text{by} \quad d, \text{one match might go up} \\
   s_2 \quad \leftarrow \quad d \quad \quad \text{by} \quad d \quad \text{M doesn't increase} \\
   \end{array} \]
   
   \( D - d(s_1, s_2) \) decreases by \( 2d \)
   
   for any other \( s, \text{say} \) right, \( d(s, s_i) \) increases by \( d \)
   
   \( d(s_1, s_2) \) decreases by \( d \)
   
   so \( \leq d(s, r) \) is constant.
   
   \( \phi \) decreases by \( \geq t_i \)

Ex. Find an example to show competitive ratio can be \( k \).
For the general k-server problem:

- \( \frac{1}{2k-1} \) [Koutsoupias & Papadimitriou 1994] combines local greedy with "retrospective alg." that goes to the state the opt. alg. would be in based on requests so far.

For randomized algs:
- lower bound \( \frac{\Omega}{\log n} \) (as for paging)
- upper bound \( O(\log k)^6 \) progress in 2018 Lee