# CS1570 — Midterm summary sheet (lectures of October 13–29)

## 1. Amortized analysis: the potential method

Assign a potential $\Phi \geq 0$ to the current state. For operation $i$ with real cost $c_i$:

$$\tilde{c}_i = c_i + \Delta\Phi_i = c_i + \Phi_{\text{after}} - \Phi_{\text{before}}.$$

Summing, $\Delta\Phi$ *telescopes*:

$$\sum_i c_i = \sum_i \tilde{c}_i - (\Phi_{\text{final}} - \Phi_{\text{initial}})$$

$$\leq \sum_i \tilde{c}_i + \Phi_{\text{initial}}.$$

So it suffices to bound each amortized cost $\tilde{c}_i$.

▶ *Designing $\Phi$: ask "when the algorithm pays a lot, what has it gained? how has the structure improved?" Expensive steps must release potential.*

## 2. List update and Move-to-Front

**Problem.** List of $n$ items, online requests $\sigma_1, \ldots, \sigma_m$. Accessing the item in position $i$ costs $i$; swapping two adjacent items costs 1. Compare with OPT, the best offline algorithm (NP-hard to compute!).

**Heuristics.** *MTF*: move the requested item to the front (cost $2k-1$ for position $k$). *Transposition*: swap with predecessor. *Frequency count (FC)*: sort by counters.

- Transposition is bad: list $A_1 \ldots A_\ell\, B\, C$, requests $C, B, C, B, \ldots$: each costs $> \ell$; MTF pays $\approx 3$.
- FC is bad: each $A_i$ requested $\ell$ times in a row: FC $= \Theta(\ell^3)$, MTF $= \Theta(\ell^2)$.

**Theorem.** For every sequence, cost(MTF) $\leq 4 \cdot$ cost(OPT).

**Potential.** $\Phi = 2\times$ (number of inversions between MTF's list and OPT's list); $\Phi_{\text{initial}} = 0$, $\Phi \geq 0$.

**Lemma.** If OPT pays $j = k^* + s$ for a request (access at $k^*$, $s$ swaps), then $\tilde{c} \leq 4j$.

**Proof idea.** Split into two phases, one list changing at a time.

- Phase 1, MTF moves $x$ from position $k$ to the front. The $k-1$ items in front of $x$ split into $P_a$ (also in front in OPT; new inversion, $+1$) and $P_b$ (behind in OPT; inversion removed, $-1$). Since $|P_a| \leq k^* - 1$, $\Delta\Phi \leq 4(k^*-1) - 2(k-1)$.
- Phase 2, each OPT swap changes $\Phi$ by $\leq 2$: $\Delta\Phi \leq 2s$.

$\tilde{c} \leq (2k-1) + 4(k^*-1) - 2(k-1) + 2s \leq 4(k^*+s)$.

▶ *The real cost $2k$ cancels against $-2k$ from the potential: MTF pays a lot only when its list was far from OPT's, and paying brings it closer.*

▶ *We compare to OPT without knowing what OPT does.*

## 3. Splay trees

**Splay($x$):** after a BST search, bring $x$ to the root by *double* rotations (zig-zig: $x, y$ both left children; zig-zag: $x$ right of $y$, $y$ left of $z$), plus possibly one final single rotation (zig). Cost = 1 per double or final rotation.

**Why not single rotations?** On a path $1, 2, \ldots, n$, lifting $n$ by single rotations leaves depth $\Theta(n)$: alternating requests cost $\Theta(n)$ each.

**Theorem.** $m$ accesses on an $n$-node splay tree cost $O(m \log n + n \log n)$.

**Potential.** $s(x)$ = size of subtree of $x$, rank $r(x) = \lfloor \log_2 s(x) \rfloor$, $\Phi = \sum_x r(x) \in [0, O(n \log n)]$. A path has $\Phi \approx n \log n$; splaying its deepest node roughly halves it.

**Access Lemma.** Amortized cost of splaying $x$ $\leq 1 + 3(r(\text{root}) - r(x)) \leq 1 + 3\log_2 n$.

**Proof idea.** Each rotation changes only the ranks of $x, y, z$ (*locality*). Show each double rotation has amortized cost $\leq 3(r'(x) - r(x))$, and the final zig $\leq 1 + 3(r'(x) - r(x))$; summing telescopes.

- Zig-zig, $r(x) = r(z) = r$: a node and its two children cannot all have rank $r$, so $r'(z) < r$ and the amortized cost is $\leq 0$.
- Zig-zig, $r(x) < r(z)$: amortized cost $\leq 1 + 2(r'(x) - r(x)) \leq 3(r'(x) - r(x))$.
- Zig-zag: similar.

▶ *Expensive accesses happen in unbalanced trees, and splaying rebalances them, decreasing $\Phi$.*

## 4. Iterated improvement

$S \leftarrow$ empty or basic solution
**while** possible: find a better $S'$; $S \leftarrow S'$
**output** $S$

Unlike greedy, $S'$ may *undo* earlier choices: you can change your mind. Key questions: (1) what is a local improvement? (2) if none exists, is $S$ optimal? (3) how fast can we find one?

## 5. Bipartite matching

$G = (X, Y, E)$, $n = |X| + |Y|$, $m = |E|$. A *matching* $M \subseteq E$ has no two edges sharing a vertex; maximize $|M|$. Greedy fails (e.g. on a 6-cycle).

**Augmenting path** for $M$: alternates between $E \setminus M$ and $M$, both endpoints unmatched (exposed). It has odd length $2k+1$ ($k+1$ edges not in $M$).

**Augment:** $M \oplus P = (M \setminus P) \cup (P \setminus M)$ is a matching with $|M| + 1$ edges.

**Berge.** $M$ is maximum $\iff$ there is no augmenting path for $M$.

**Proof ($\Leftarrow$):** if $|M'| > |M|$, then $H = M \oplus M'$ has max degree 2, so it is a union of paths and even cycles with alternating edges. Some component has more $M'$-edges than $M$-edges; it is an augmenting path for $M$.

▶ *Symmetric difference with a better solution exhibits an improving move.*

**Finding a path.** Directed graph $D_M$: edges of $E \setminus M$ go $X \to Y$, edges of $M$ go $Y \to X$. An augmenting path is a directed path from an exposed $x \in X$ to an exposed $y \in Y$: BFS/DFS from all exposed $X$ vertices at once (add a super-source $s$).

**Algorithm:** $\leq n$ augmentations $\times O(m) = O(nm)$.

▶ *Turn "find an improvement" into a reachability problem.*

## 6. Hopcroft–Karp: $O(m\sqrt{n})$

**Fact.** If $|N| = |M| + s$ (both matchings), then $M \oplus N$ contains $\geq s$ vertex-disjoint augmenting paths for $M$.

**Phase:** $k \leftarrow$ length of a shortest augmenting path; find a *maximal* set of vertex-disjoint augmenting paths of length $k$; augment along all of them.

**Lemma 1.** $P$ shortest for $M$, $P'$ augmenting for $M \oplus P \Rightarrow |P'| \geq |P| + 2|P \cap P'|$ (common edges). Proof: $M \oplus M'' = P \oplus P'$ contains 2 disjoint augmenting paths for $M$, each of length $\geq |P|$.

**Lemma 2.** $O(\sqrt{n})$ phases.

- The shortest augmenting length strictly increases each phase. If $P'$ avoids all $P_i$, maximality forces $|P'| > k$. Otherwise, flip $P_1, \ldots, P_r$ one by one, with the path met by $P'$ last, and apply Lemma 1 to that last step; $P'$ shares an edge with it (the matching edge at the shared vertex).
- After $\sqrt{n}$ phases, all augmenting paths have length $\geq \sqrt{n}$. $M \oplus M^*$ has $|M^*| - |M|$ disjoint such paths, so $|M^*| - |M| \leq \sqrt{n}$: at most $\sqrt{n}$ more phases.

**Lemma 3.** One phase in $O(m)$: BFS layers from exposed $X$ up to the first level with an exposed $Y$; keep the edges between consecutive levels; repeatedly trace a path back from an exposed $y$; delete its vertices, and propagate deletions via predecessor counters (deletion queue).

▶ *Augment along shortest paths, many at once; the shortest length is a progress measure.*

## 7. König's theorem

**Vertex cover:** a set of vertices touching every edge.

**Theorem.** In a bipartite graph, max matching = min vertex cover. (False in general: the triangle.)

**Proof.** ($\geq$) each edge of $M^*$ needs its own cover vertex. ($\leq$) let $L$ = vertices reachable from exposed $X$ vertices by alternating paths; then $C = (X \setminus L) \cup (Y \cap L)$.

- There is no edge from $X \cap L$ to $Y \setminus L$, so $C$ is a cover.
- Each vertex of $C$ is matched (Berge), and no edge of $M^*$ has both ends in $C$, so $|C| \leq |M^*|$.

▶ *The algorithm's final search produces a certificate of optimality.*

## 8. Hall's theorem

There is a matching of size $|X| \iff |N(S)| \geq |S|$ for all $S \subseteq X$.

**Proof ($\Leftarrow$), induction on $|X|$.**

- No tight set ($|N(S)| \geq |S| + 1$ for all $\emptyset \neq S \subsetneq X$): match any edge $xy$ and remove $x, y$; the condition still holds.
- Tight set $S$ ($|N(S)| = |S|$, $T = N(S)$): the condition holds in $G[S \cup T]$ and in $G[(X \setminus S) \cup (Y \setminus T)]$; recurse on both and combine.

It also follows from König (take $S = X \setminus C$).

## 9. Network flows

**Network:** directed $G$, source $s$, sink $t$, capacities $c(e) \geq 0$. **Flow:** $0 \leq f(e) \leq c(e)$, and conservation (in = out) at every $v \neq s, t$. **Value** $|f|$ = net flow out of $s$.

▶ *Atomic objects: $s$–$t$ paths preserve conservation.* Greedy path-pushing gets stuck; we must be able to undo flow.

**Residual graph $G_f$:** a forward edge $uv$ with capacity $c - f$ if $f < c$; a backward edge $vu$ with capacity $f$ if $f > 0$. **Augmenting path:** an $s$–$t$ path in $G_f$; push its bottleneck $\delta$: $|f|$ increases by $\delta$.

**Ford–Fulkerson:** $f \leftarrow 0$; while $G_f$ has an augmenting path, augment.

**Cut** $(S, T)$, $s \in S$, $t \in T$: $\text{cap}(S,T) = \sum_{u \in S, v \in T} c(uv)$.

**Lemma.** $|f| = f(S \to T) - f(T \to S) \leq \text{cap}(S, T)$ for every cut.

**Max-flow min-cut.** The following are equivalent: (1) $f$ is maximum; (2) there is no augmenting path; (3) $|f| = \text{cap}(S, T)$ for some cut. Hence $\max |f| = \min \text{cap}$.

**Proof (2)⇒(3):** let $S$ = vertices reachable from $s$ in $G_f$. Edges $S \to T$ are saturated and edges $T \to S$ carry 0.

**Corollary.** With integer capacities, FF terminates with a max flow and a min cut.

▶ *Same pattern as matchings: iterated improvement + augmenting paths; when stuck, the reachable set is a certificate (min cut ↔ vertex cover).*

## Recurring methods

- **Potential functions:** amortize, telescope, compare to OPT.
- **Split a complex step into phases** or atomic steps, and analyse one change at a time.
- **Iterated improvement:** define a local move; prove "no move ⇒ optimal".
- **Exchange via $\oplus$:** compare the current solution with a better one.
- **Reduce to graph search:** build a directed auxiliary graph, then BFS/DFS.
- **Certificates of optimality:** vertex cover, Hall set, min cut.
- **Bad examples** show why a naive rule fails (FC, transposition, single rotations, greedy).