Stable Marriages: An Introductory Example (1)

Given:
- $n$ women $w_1, w_2, \ldots, w_n$
- $n$ men $m_1, m_2, \ldots, m_n$
- $n$ marriages $(w_{i_1}, m_{j_1}), (w_{i_2}, m_{j_2}), \ldots, (w_{i_n}, m_{j_n})$
PROPOSAL-ALGORITHM($M$, $W$)
1  while there is an unmarried man $m$
2    do $m$ chooses his favourite woman $w$ he has not proposed to yet
3       $m$ proposes to $w$
4       if $w$ is not married or likes $m$ better than her current partner $m'$
5          then $w$ divorces $m'$
6          $w$ marries $m$

- Is there always a set of $n$ stable marriages?
- Does the algorithm ever terminate?
- Does the algorithm always produce a correct answer?
- How efficient is the algorithm? Can we give an upper bound on its running time?
**Linear Time (1)**

**Example:** Merging two sorted sequences

- **Base case:** \(n=0\), \(k=1\)
- IH: first \(k\) elements are sorted and smaller than anything in \(A \cup B\)

1. Assume \(A_{i:j} \subset C_{b:c}\)
2. \(A_{i:j} < A_{s-j} \forall s > i\) since \(A\) is sorted
3. \(A_{i:j} < B_{k:j} \forall k \leq B\) since \(B\) is sorted and \(A_{i:j} \subset C_{b:c}\)

By IH: \(C_{e:j} < A_{i:j} \forall e\)

IH holds
**Linear Time (2)**

**MERGE**\((A, B)\)

1.  \(C \leftarrow \emptyset\)
2.  **while** \(A\) and \(B\) are non-empty
3.   **do** Let \(a\) be the first element in \(A\)
4.   Let \(b\) be the first element in \(B\)
5.   **if** \(a < b\)
6.      **then** Remove \(a\) from \(A\)
7.      Append \(a\) to \(C\)
8.   **else** Remove \(b\) from \(B\)
9.      Append \(b\) to \(C\)
10.  **if** \(A\) is non-empty
11.     **then** Append \(A\) to \(C\)
12.     **else** Append \(B\) to \(C\)
13.  **return** \(C\)
**Linear Time (3)**

**MERGE**\((A, p, q, r)\)

1. \(n_1 \leftarrow q - p + 1\)
2. \(n_2 \leftarrow r - q\)
3. **for** \(i \leftarrow 1\) **to** \(n_1\)
   4. **do** \(L[i] \leftarrow A[p + i - 1]\)
5. **for** \(i \leftarrow 1\) **to** \(n_2\)
6. **do** \(R[i] \leftarrow A[q + i]\)
7. \(L[n_1 + 1] \leftarrow \infty\)
8. \(R[n_2 + 1] \leftarrow \infty\)
9. \(i \leftarrow 1\)
10. \(j \leftarrow 1\)
11. **for** \(k \leftarrow 1\) **to** \(r - p + 1\)
12. **do if** \(L[i] < R[j]\)
13. **then** \(A[k] \leftarrow L[i]\)
14. \(i \leftarrow i + 1\)
15. **else** \(A[k] \leftarrow R[j]\)
16. \(j \leftarrow j + 1\)
**Example:** Closest pair

Given a set $P$ of $n$ points, find the two points in $P$ whose distance is smallest.

---

Proof by contradiction:

Assumption: $p_i$ and $q_j$ are not the smallest distance.

- $p_iq_j$ are closer together.

But, we look at all pairs of points so we would have chosen $p_xq_y$.

$\therefore$ contradiction!
**Quadratic Time (2)**

\[
\text{CLOSESTPAIR}(P)
\]

1. \( d \leftarrow +\infty \)
2. \textbf{for} every point \( p \in P \)
3. \textbf{do for} every point \( q \in P \) such that \( q \neq p \)
4. \quad \textbf{do if} \( \text{dist}(p, q) < d \)
5. \qquad \textbf{then} \( d \leftarrow \text{dist}(p, q) \)
6. \qquad \text{pair} \leftarrow (p, q)
7. \textbf{return} \text{pair}
**Quadratic Time (2)**

**CLOSESTPAIR**\((P)\)

1. \(d \leftarrow +\infty\)
2. **for** every point \(p \in P\)
3. **do for** every point \(q \in P\) such that \(q \neq p\)
4. **do if** \(\text{dist}(p, q) < d\)
5. **then** \(d \leftarrow \text{dist}(p, q)\)
6. **pair** \(\leftarrow (p, q)\)
7. **return** pair

**Comments:**
- Quadratic time is easy to recognize here because there are two nested loops iterating over the whole input.
- Clever use of divide-and-conquer and geometric insight leads to an \(O(n \log n)\) solution.