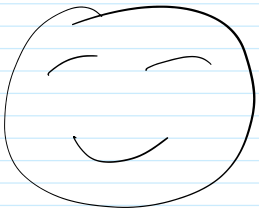
Algorithms Description

Monday, January 13, 2025 12:15 PM

Previously

- Syllabus
- Closest Pair → numbers, points
- HW0 assigned (due Today!) → midnight



Today

1. Describing Algorithms
2. Two Examples

Algorithm — a specific series of steps defined to solve a problem

Computation problems — a problem which can be solved through logic (computable)

Algorithms are methods
↳ independent of specific programming languages.

Describing Algorithms

(1) Describe the purpose
Describe the steps

(2) Justify the correctness.
↳ proof/claim.

(3) Efficiency. → running time.
↳ counting the number of steps.

Note

① purpose → good place to nail down definitions.

steps

- pseudocode → too detailed, no insight.
- English prose → sometimes vague.
  → algorithms usually have structure.

mix of these

Example 1 (closest numbers algorithm)

Given a list L of numbers, the algorithm finds the min. difference between any two distinct elements $x, y \in L$.

Algorithm

$\Theta(n \log n)$

① Sort list L in ascending order with merge sort.

② Loop through each adjacent pair in L, compute their difference while storing the min. difference encountered. $|L| = n$

③ Output the stored min. difference.

Correctness

Claim: The min. difference is between some adjacent pair.

↳ proof via contradiction.

Efficiency

Efficiency

Step ① is the most significant and takes $\Theta(n \log n)$

The algorithm is $\Theta(n \log n)$

Example 2

$5628 \times 726 = ?$

```
   5628
 x  726
-------
  33768
  11256
 39396
-------
4085928
```

lattice multiplication

↳ ~1400 textbook.

Input: Two numbers X, Y represented by $X[1] \ldots X[m]$ and $Y[1] \ldots Y[n]$

Output: The product of X and Y.

---

subroutines

① addition

② single-digit multiplication ⟶ look-up table.

③ mult. by powers of 10 (shift)

Steps

```
prod ← 0
for i from m down to 1:
    for j from n down to 1:
```

Steps

prod ← 0

for i from m down to 1:

  for j from n down to 1:

    prod += $X[i] \cdot y[j] \cdot 10^{(m-i)+(n-j)}$

return prod.

Efficiency

$O(mn)$

m and n are the length of the numbers

Peasant Multiplication ~1650 BCE.

↳ early computers.

subroutine

① addition

② parity

③ duplation → doubling

④ mediation → halving, round down.

mult(x, y)

prod ← 0

while x > 0:

  if x is odd:

    prod ← prod + y

$$x \cdot y = \begin{cases} 0 & \text{if } x = 0 \\ \lfloor x/2 \rfloor \cdot 2y & \text{if } x \text{ is even} \\ | x / | & \text{if } x \end{cases}$$

prod ← prod + y

x ← ⌊x/2⌋

y ← y + y

return prod.

$\lfloor x/2 \rfloor \cdot 2y + y$ if x is odd

Efficiency:

$O(\log x \cdot \log y)$

$O(m \; n)$

$\log x \quad \log y$