3-SUM Hardness

Def: (3-SUM)

Given: Sets A, B, C of n integers $\quad A, B, C \subseteq \{-U, \dots, U\}$

Task: Decide if there are $a \in A, b \in B, c \in C$ s.t. $a+b+c=0$

Equivalent: $a+b=c$

$\exists a,b,c \in X$ s.t. $a+b+c=0$ et c

(Let $X = \{a+4U \mid a \in A\} \cup B \cup \{c-4U \mid c \in C\}$

Algorithms:

- $O(n^3)$ (trivial)
- $O(n^2)$ (textbook)
- $O(n + U \operatorname{polylog}(U))$
- $O\left(n^2 \frac{(\log\log n)^2}{\log^2 n}\right)$
- $O\left(n^2 \frac{\log^2 w}{w}\right)$

[Baran, Demaine, Patrascu '05] using Word RAM "bit tricks" cell size $w = \Omega(\log n)$

- $O\left(n^2 \cdot \frac{(\log\log n)^2}{\log n}\right)$ [Grønlund, Pettie '14] no "bit tricks"

Quadratic algorithm: $\exists a,b,c \in A$ s.t. $a+b+c=0$

Sort A in increasing order $A = \{a_1, \dots, a_n\}$

For each $a \in A$ check if $\exists a,b \in A$ s.t. $a+b+c=0$

initialize $i=n, j=1$

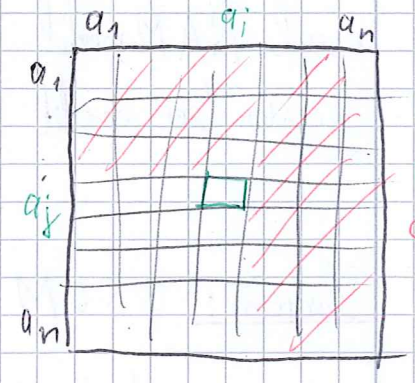
while $i>0$ and $j \le n$:

if $a_i + a_j = -c$: return (a_i, a_j, c)

if $a_i + a_j > -c$: $i = i-1$

otherwise $j = j+1$

return no solution



Invariant: red area cannot give desired pair

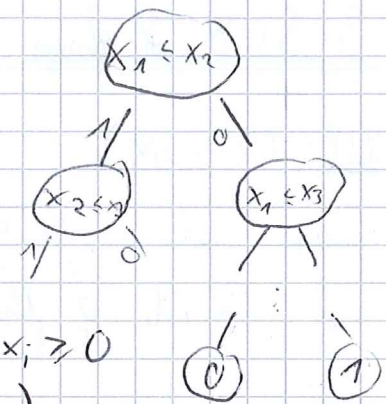
Running time: $O(n)$ per $a \in A$

$\to O(n^2)$ for loop + $O(n \log n)$ for sorting $\to O(n^2)$

Decision Tree Complexity

Problem P on input $x_1, \dots, x_n$

Decision Tree:



- each inner node is a comparison: $x_i \le x_j$
  more generally: linear combination: $\sum a_i x_i \ge 0$
- outgoing edges are labelled 1/0 (true/false)
- leaf determines result (e.g. 0/1)

Def: Decision tree complexity of problem P is the minimum depth of any decision tree for P

→ Lower bound for running time of any algorithm (that uses only additions and comparisons, no "bit tricks")

Compare: any decision tree for sorting n numbers has depth $\Omega(\log n)$

Alternative interpretation:

RAM with 2 types of cells:

	special	standard
Stores:	input numbers & intermediate results	$O(\log n)$ bit numbers
Operations:	add, subtract, compare (result stored as comparison may be stored in standard cell)	all standard arithmetic & logical operations and comparisons

usual RAM model: each operation takes constant time

decision tree model: comparisons of special numbers cost 1
all other operations are for free

Theorem: 3SUM has a decision tree of depth $O(n^{3/2} \log n)$

→ no quadratic lower bound for 3SUM in decision tree model

$\exists a,b,c \in A$ s.t. $a+b+c=0$?

Step 1:

- sort A in increasing order — $O(n \log n)$ comparisons
- partition A into $\frac{n}{g}$ groups $A_1, \dots, A_{n/g}$ of size g each — no comparisons
  ($\max A_i \le \min A_{i+1}$) (write $A_i = \{a_{i,1}, \dots, a_{i,g}\}$)
- sort $D := \bigcup_{i=1}^{n/g} (A_i - A_i) = \{a-b \mid \exists\, i\ a,b \in A_i\}$ — $O(|D| \log |D|)$ comparisons $= O(ng \log(ng))$
- for all i,i': sort $A_{i,i'} := A_i + A_{i'} = \{a+b \mid a \in A_i, b \in A_{i'}\}$ — no comparisons!

Sorting D: build list L_D containing all (i,j,k) with
$i \in \{1, \dots, n/g\}$, $j,k \in \{1, \dots, g\}$
sorted by $a_{ij} - a_{ik}$ ascendingly

$\Rightarrow$ can compare any $a_{ij} - a_{ik}$ and $a_{i'j'} - a_{i'k'}$ without comparison operation

Fredman's trick: $a_{ij} + a_{i'j'} \le a_{ik} + a_{i'k'}$

$\Leftrightarrow a_{i'j'} - a_{i'k'} \le a_{ik} - a_{ij}$

$\Leftrightarrow (i',j',k')$ appears before (i,k,j) in L_D

Can compare any $a_{ij} + a_{i'j'}$ and $a_{ik} + a_{i'k'}$ with comp. op.

any numbers in $A_{i,i'} = A_i + A_{i'}$

Step 2:

for each $c \in A$: (Goal: check if $\exists\, a,b \in A$ s.t. $a+b+c=0$) — n iterations

initialize $i = n/g$, $j = 1$

while $i > 0$ and $j \le \frac{n}{g}$ — $O(n/g)$ iterations

if $-c \in A_j$: return "solution found" — $O(\log(g^3)) = O(\log n)$ comparisons using binary search

if $\min(A_i) + \max(A_j) > -c$: $i = i-1$

otherwise $j = j+1$

Total: $O((ng + n^2/g) \log n)$ comparisons
$= O(n^{3/2} \log n)$ for $g = \sqrt{n}$

return "no solution"

Theorem: 3SUM can be solved in time $O\left(n^2 \frac{\text{poly}\log\log n}{\sqrt{\log n}}\right)$

(without "bit tricks")

Expensive step:

"For all i,i': sort $A_{i,i'} = A_i + A_{i'} = \{a+b \mid a \in A_i, b \in A_{i'}\}$

will take $O((n/g)^2 \cdot g^2 \log(g^2))$ time

$\rightarrow$ total time $O(n^2 \log(g^2))$

Faster implementation of this step:

make each $A_{i,i'}$ totally ordered:

replace A_i by $\{a_{ij} \cdot (2g)^2 + j \mid 1 \le j \le g\}$

replace $A_{i'}$ by $\{a_{i'j} \cdot (2g)^2 + j \cdot (2g) \mid 1 \le j \le g\}$

$\Rightarrow$ no $a \in A_i$, $b \in A_{i'}$ and $a' \in A_i$, $b' \in A_{i'}$ sum up to the same value

and: from "new" $A_i + A_{i'}$ we can recover "old" $A_i + A_{i'}$

Consider any permutation $P = ((\pi_1, \sigma_1), (\pi_2, \sigma_2), \dots, (\pi_{g^2}, \sigma_{g^2}))$ of $\{1,\dots,g\} \times \{1,\dots,g\}$

P corresponds to this ordering of $A_{i,i'}$:

$(a_{i,\pi_1} + a_{i',\sigma_1}, a_{i,\pi_2} + a_{i',\sigma_2}, \dots, a_{i,\pi_n} + a_{i',\sigma_n})$

This is the sorted ordering of $A_{i,i'}$ iff

$$a_{i\pi_k} + a_{i'\sigma_k} < a_{i\pi_{k+1}} + a_{i'\sigma_{k+1}} \quad \forall\, 1 \le k \le g^2 - 1$$

$$\Leftrightarrow a_{i\sigma_k} - a_{i'\sigma_{k+1}} < a_{i\pi_{k+1}} - a_{i\pi_k} \quad \text{(Fredman's trick)}$$

Vector notation:

$$\begin{pmatrix} a_{i\sigma_1} - a_{i'\sigma_2} \\ \vdots \\ a_{i\sigma_{g^2-1}} - a_{i'\sigma_{g^2}} \end{pmatrix} < \begin{pmatrix} a_{i\pi_2} - a_{i\pi_1} \\ \vdots \\ a_{i,g^2} - a_{i,g^2-1} \end{pmatrix}$$

vector x dominates vector y if $x_i < y_i$ for all coordinates i

Def. (Dominance Reporting)
Given: Sets A, B of vectors in $\mathbb{Z}^d$, $|A| + |B| = m$
Task: report all pairs $a \in A, b \in B$ where b dominates a

Thm: Dominance Reporting can be solved in time
$O(m (\log m)^d + \text{outputsize})$

Algorithm: "For all i, i': sort $A_{i,i'} = A_i + A_{i'}$"

For each permutation $P = ((\pi_1, \sigma_1), \dots, (\pi_{g^2}, \sigma_{g^2}))$ of $\{1..g\} \times \{1..g\}$
construct sets $X = \{ (a_{i'}, \sigma_k - a_{i'} \omega_{k+1})_{1 \le k \le g^2-1} \mid 1 \le i' \le n/g \}$

$$Y = \{ (a_i, \pi_{k+1} - a_i \pi_k)_{1 \le k \le g^2-1} \mid 1 \le i \le n/g \}$$

solve dominance reporting on X, Y
for each reported pair (i, i')
sorted ordering of $A_{i,i'}$ is given by P:
$A_{i,i'} = (a_i \pi_1 + a_{i'} \sigma_1 \quad a_i \pi_2 + a_{i'} \sigma_2 \quad \dots \quad a_i \pi_n + a_{i'} \sigma_n)$

Analysis:
Total output size over all permutations: $(n/g)^2$
$\Rightarrow$ time for sorting all $A_{i,i'}$:

$$O\left((g^2)! \cdot \left(\frac{n}{g}\right) (\log(n/g))^{g^2} + (n/g)^2 \right) = O((n/g)^2)$$

setting $g = 0.1 \cdot \sqrt{\log n / \log\log n}$

$$(g^2)! \le (g^2)^{g^2} \le (\log n)^{g^2} = (\log n)^{(0.01 \log n)/\log\log n} = n^{0.01}$$

$$(\log n/g)^{g^2} \le (\log n)^{g^2} \le n^{0.01}$$

Total time: $O(n^2 \log(g)/g)$

$$= O\left(n^2 \frac{\text{poly} \log\log n}{\sqrt{\log n}}\right)$$