3SUM

given sets A, B, C of n integers

are there $a \in A, b \in B, c \in C$ such that $a + b + c = 0$?

(we assume that we can add/subtract/compare input integers in constant time)

trivial algorithm: $O(n^3)$

well-known: $O(n^2)$

Conjecture: no $O(n^{2-\varepsilon})$ algorithm

→ 3SUM-Hardness [Gajentaan,Overmars'95]

More Known Algorithms

trivial: $O(n^3)$

well-known: $O(n^2)$

using FFT: $O(n + U \operatorname{polylog} U)$ for numbers in $\{-U, \ldots, U\}$

using Word RAM bit-tricks: $O(n^2 \cdot \frac{\log^2 w}{w})$, $O(n^2 \cdot \frac{(\log\log n)^2}{\log^2 n})$
(cell size $w = \Omega(\log n)$,
each number fits in a cell) [Baran,Demaine,Patrascu'05]

no bit-tricks: $O(n^2 \cdot \frac{(\log\log n)^2}{\log n})$ [Gronlund,Pettie'14]

we prove a simplified version:

Thm: Without bit-tricks, 3SUM is in time $O(n^2 \cdot \frac{\operatorname{poly}\log\log n}{\sqrt{\log n}})$

Equivalent Variants

1) given sets A, B, C of n integers
are there $a \in A, b \in B, c \in C$ such that $a + b + c = 0$?

2) given sets A, B, C of n integers — replace C by $\{-c \mid c \in C\}$
are there $a \in A, b \in B, c \in C$ such that $a + b = c$?
$\Leftrightarrow a + b - c = 0$

3) given sets A, B, C of n integers and target t — replace C by $\{c - t \mid c \in C\}$
are there $a \in A, b \in B, c \in C$ such that $a + b + c = t$?
$\Leftrightarrow a + b + (c - t) = 0$

4) given a set X of n integers
are there $x, y, z \in X$ such that $x + y + z = 0$?

↑: set $A, B, C := X$

↓: set $X := \{a + 4U \mid a \in A\} \cup B \cup \{c - 4U \mid c \in C\}$

where $A, B, C \subseteq \{-U, .., U\}$

Outline

1) algorithm for small universe

2) quadratic algorithm

3) small decision tree

4) logfactor improvement

5) some 3SUM-hardness results

Algorithm for Small Numbers

$O(n + U \operatorname{polylog} U)$ for numbers in $\{-U, \ldots, U\}$

add U to each number, then numbers are in $\{0, .., 2U\}$ and
we want $a \in A, b \in B, c \in C$ such that $a + b + c = 3U$

define polynomials $p_A(x) \coloneqq \sum_{a \in A} x^a$ and similarly $p_B(x)$, $p_C(x)$
have degree at most $2U$

compute $q(x) \coloneqq p_A(x) \cdot p_B(x) \cdot p_C(x) = (\sum_{a \in A} x^a)(\sum_{b \in B} x^b)(\sum_{c \in C} x^c)$

what is the coefficient of x^{3U} in $q(x)$? $\quad (x^a \cdot x^b \cdot x^c = x^{a+b+c})$

it is the number of (a, b, c) summing to $3U$

use efficient polynomial multiplication (via Fast Fourier Transform):

polynomials of degree d can be multiplied in time $O(d \operatorname{polylog} d)$

max planck institut informatik

Outline

1) algorithm for small universe

2) quadratic algorithm

3) small decision tree

4) logfactor improvement

5) some 3SUM-hardness results

max planck institut informatik

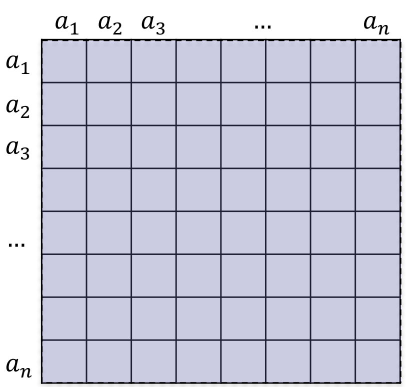
Quadratic Algorithm

given a set A of n integers
are there $a, b, c \in A$ such that $a + b + c = 0$?

sort A in increasing order: $A = \{a_1, \ldots, a_n\}$

for each $c \in A$: *check whether there are $a, b \in A$ s.t. $a + b + c = 0$*

initialize $i = n, j = 1$
while $i > 0$ and $j \leq n$:
  if $a_i + a_j = -c$: return (a_i, a_j, c)
  if $a_i + a_j > -c$: $i \coloneqq i - 1$
  if $a_i + a_j < -c$: $j \coloneqq j + 1$
return "no solution"



max planck institut informatik

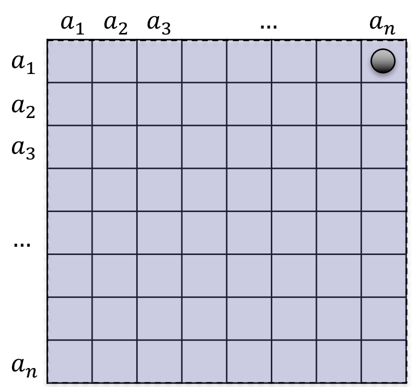
Quadratic Algorithm

given a set A of n integers
are there $a, b, c \in A$ such that $a + b + c = 0$?

sort A in increasing order: $A = \{a_1, \ldots, a_n\}$

for each $c \in A$: *check whether there are $a, b \in A$ s.t. $a + b + c = 0$*

initialize $i = n, j = 1$
while $i > 0$ and $j \leq n$:
  if $a_i + a_j = -c$: return (a_i, a_j, c)
  if $a_i + a_j > -c$: $i \coloneqq i - 1$
  if $a_i + a_j < -c$: $j \coloneqq j + 1$
return "no solution"



max planck institut informatik

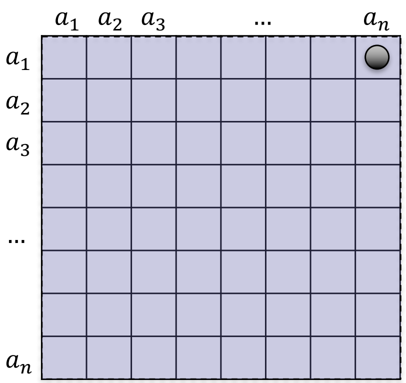
Quadratic Algorithm

given a set A of n integers
are there $a, b, c \in A$ such that $a + b + c = 0$?

sort A in increasing order: $A = \{a_1, \ldots, a_n\}$

for each $c \in A$: *check whether there are* $a, b \in A$ *s.t.* $a + b + c = 0$

initialize $i = n, j = 1$
while $i > 0$ and $j \leq n$:
  if $a_i + a_j = -c$: return (a_i, a_j, c)
  if $a_i + a_j > -c$: $i \coloneqq i - 1$
  if $a_i + a_j < -c$: $j \coloneqq j + 1$
return "no solution"



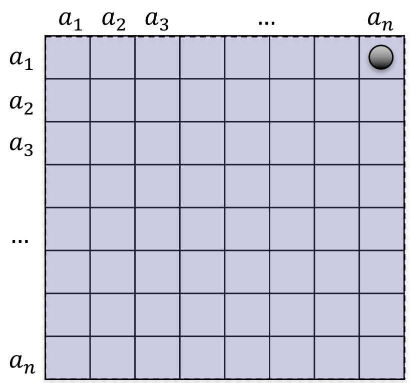
Quadratic Algorithm

given a set A of n integers
are there $a, b, c \in A$ such that $a + b + c = 0$?

sort A in increasing order: $A = \{a_1, \ldots, a_n\}$

for each $c \in A$: *check whether there are* $a, b \in A$ *s.t.* $a + b + c = 0$

initialize $i = n, j = 1$
while $i > 0$ and $j \leq n$:
  if $a_i + a_j = -c$: return (a_i, a_j, c)
  if $a_i + a_j > -c$: $i \coloneqq i - 1$
  if $a_i + a_j < -c$: $j \coloneqq j + 1$
return "no solution"



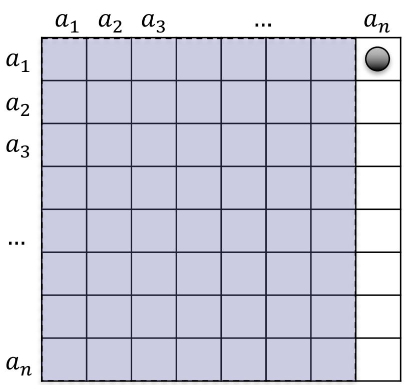
Quadratic Algorithm

given a set A of n integers
are there $a, b, c \in A$ such that $a + b + c = 0$?

sort A in increasing order: $A = \{a_1, \ldots, a_n\}$

for each $c \in A$: *check whether there are* $a, b \in A$ *s.t.* $a + b + c = 0$

initialize $i = n, j = 1$
while $i > 0$ and $j \leq n$:
  if $a_i + a_j = -c$: return (a_i, a_j, c)
  if $a_i + a_j > -c$: $i \coloneqq i - 1$
  if $a_i + a_j < -c$: $j \coloneqq j + 1$
return "no solution"



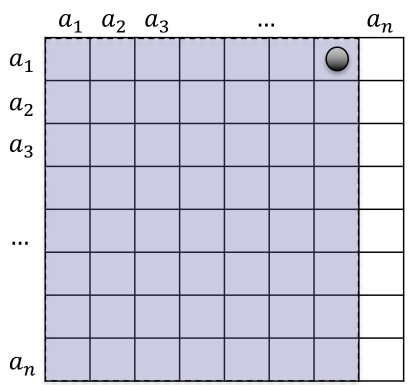
Quadratic Algorithm

given a set A of n integers
are there $a, b, c \in A$ such that $a + b + c = 0$?

sort A in increasing order: $A = \{a_1, \ldots, a_n\}$

for each $c \in A$: *check whether there are* $a, b \in A$ *s.t.* $a + b + c = 0$

initialize $i = n, j = 1$
while $i > 0$ and $j \leq n$:
  if $a_i + a_j = -c$: return (a_i, a_j, c)
  if $a_i + a_j > -c$: $i \coloneqq i - 1$
  if $a_i + a_j < -c$: $j \coloneqq j + 1$
return "no solution"



Quadratic Algorithm

given a set A of n integers
are there $a, b, c \in A$ such that $a + b + c = 0$?

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

for each $c \in A$: *check whether there are* $a, b \in A$ *s.t.* $a + b + c = 0$

initialize $i = n, j = 1$

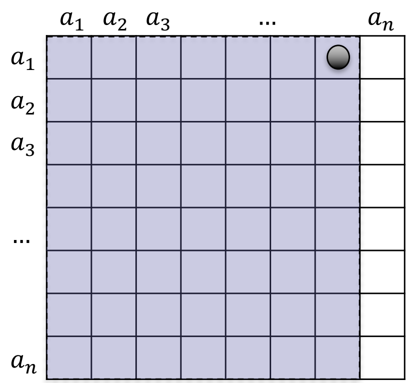
while $i > 0$ and $j \le n$:

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

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

if $a_i + a_j < -c$: $j := j + 1$

return "no solution"



max planck institut
informatik

Quadratic Algorithm

given a set A of n integers
are there $a, b, c \in A$ such that $a + b + c = 0$?

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

for each $c \in A$: *check whether there are* $a, b \in A$ *s.t.* $a + b + c = 0$

initialize $i = n, j = 1$

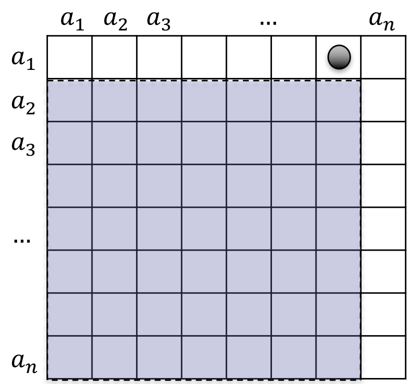
while $i > 0$ and $j \le n$:

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

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

if $a_i + a_j < -c$: $j := j + 1$

return "no solution"



max planck institut
informatik

Quadratic Algorithm

given a set A of n integers
are there $a, b, c \in A$ such that $a + b + c = 0$?

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

for each $c \in A$: *check whether there are* $a, b \in A$ *s.t.* $a + b + c = 0$

initialize $i = n, j = 1$

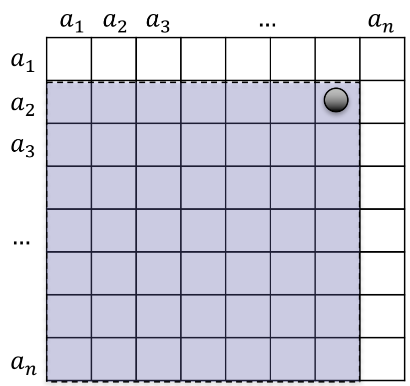
while $i > 0$ and $j \le n$:

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

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

if $a_i + a_j < -c$: $j := j + 1$

return "no solution"



max planck institut
informatik

Quadratic Algorithm

given a set A of n integers
are there $a, b, c \in A$ such that $a + b + c = 0$?

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

for each $c \in A$: *check whether there are* $a, b \in A$ *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)

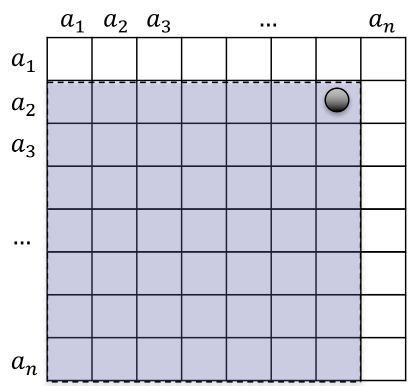
if $a_i + a_j > -c$: $i := i - 1$

otherwise: $j := j + 1$

return "no solution"

time $O(n)$ per $c \in A$

time $O(n^2)$ overall



max planck institut
informatik

Outline

1) algorithm for small universe

2) quadratic algorithm

3) small decision tree

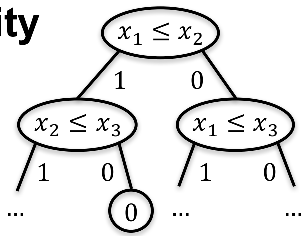
4) logfactor improvement

5) some 3SUM-hardness results

Decision Tree Complexity

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

Decision Tree Complexity



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

Decision Tree:

each **inner node** is a **comparison**: $x_i \leq x_j$

more generally any linear combination: $\sum_i \alpha_i x_i \geq 0$

outgoing edges are labeled 1/0 = true/false

all instances reaching the same **leaf** have the same result $P(x_1, \dots, x_n)$

decision tree complexity of P = minimal depth of any decision tree for P

yields a **lower bound for running time** of any algorithm
(that uses only comparisons, no bit-tricks)

where you have seen this:

Thm: Any decision tree for Sorting n numbers has depth $\Omega(n \log n)$

Thm: Any comparison-based Sorting algorithm takes time $\Omega(n \log n)$

Decision Tree Complexity

alternative interpretation:

think of $x_1, \dots, x_n$ as physical entities

we can perform **experiments**:

„experiment" or „costly comparison"

we may specify factors α_i

the outcome of the experiment tells us whether $\sum_i \alpha_i x_i \geq 0$

experiments are very *costly*, computation is *cheap*

what is the **minimal number of experiments** to decide $P(x_1, \dots, x_n)$?

= decision tree complexity

Decision Tree Complexity

alternative interpretation II:

RAM with two types of cells: **special** and **standard**

input numbers $x_1, \dots, x_n$ are stored in special cells

	special	standard
Stores:	e.g. real number	$O(\log n)$ bit number
Operations:	add, subtract, compare (result of comparison can be stored in standard cell)	all standard arithmetic and logical operations and comparisons

usual RAM cost model: each operations takes constant time

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

Decision Tree Complexity

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

why study decision tree *upper bounds*?

rules out quadratic lower bound in decision tree model

often small decision trees yield lower order improvements

Thm: Without bit-tricks, 3SUM is in time $O(n^2 \cdot \frac{\text{poly} \log \log n}{\sqrt{\log n}})$

Small Decision Tree

given a set A of n integers, are there $a, b, c \in A$ such that $a + b + c = 0$?

sort A in increasing order — $O(n \log n)$ comparisons

partition A into n/g groups: $A_1, \dots, A_{n/g}$ — write $A_i = \{a_{i,1}, \dots, a_{i,g}\}$
(all elements of A_i are smaller than all elements of A_{i+1})

sort $D \coloneqq \bigcup_{i=1}^{n/g} A_i - A_i = \{a - b \mid \exists i: a, b \in A_i\}$ — $O(|D| \log |D|) = O(ng \log(ng))$ comparisons

i.e., build a list L_D containing all (i, j, k) with $i \in \{1, \dots, n/g\}, j, k \in \{1, \dots, g\}$
sorted by $a_{i,j} - a_{i,k}$ ascendingly

this preprocessing allows to compare any $a_{i,j} - a_{i,k}$ and $a_{i',j'} - a_{i',k'}$ without any costly comparisons

Fredman's trick: $a_{i,j} + a_{i',j'} \leq a_{i,k} + a_{i',k'} \iff a_{i',j'} - a_{i',k'} \leq a_{i,k} - a_{i,j}$

so this preprocessing allows to compare any $a_{i,j} + a_{i',j'}$ and $a_{i,k} + a_{i',k'}$ without any costly comparisons:

$a_{i,j} + a_{i',j'} \leq a_{i,k} + a_{i',k'} \iff (i', j', k')$ appears before (i, k, j) in L_D

any numbers in $A_{i,i'} \coloneqq A_i + A_{i'} = \{a + b \mid a \in A_i, b \in A_{i'}\}$

Small Decision Tree

given a set A of n integers, are there $a, b, c \in A$ such that $a + b + c = 0$?

sort A in increasing order — $O(n \log n)$ comparisons

partition A into n/g groups: $A_1, \dots, A_{n/g}$
(all elements of A_i are smaller than all elements of A_{i+1})

sort $D \coloneqq \bigcup_{i=1}^{n/g} A_i - A_i = \{a - b \mid \exists i: a, b \in A_i\}$ — $O(|D| \log |D|) = O(ng \log(ng))$ comparisons

for all i, i': sort $A_{i,i'} \coloneqq A_i + A_{i'} = \{a + b \mid a \in A_i, b \in A_{i'}\}$ — no comparisons!

for each $c \in A$: *check whether there are* $a, b \in A$ *s.t.* $a + b + c = 0$

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

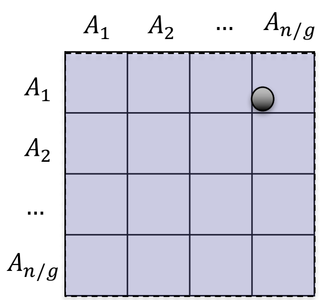
while $i > 0$ and $j \leq n/g$:

if $-c \in A_{i,j}$: return "solution found"

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

otherwise: $j \coloneqq j + 1$

return "no solution"



Small Decision Tree

given a set A of n integers, are there $a, b, c \in A$ such that $a + b + c = 0$?

sort A in increasing order — $O(n \log n)$ comparisons

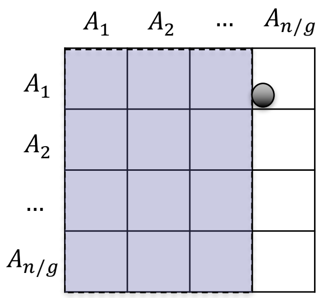
partition A into n/g groups: $A_1, \dots, A_{n/g}$
(all elements of A_i are smaller than all elements of A_{i+1})

sort $D := \bigcup_{i=1}^{n/g} A_i - A_i = \{a - b \mid \exists i: a, b \in A_i\}$ — $O(|D| \log |D|) = O(ng \log(ng))$ comparisons

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!

for each $c \in A$: *check whether there are $a, b \in A$ s.t. $a + b + c = 0$*

- initialize $i = n/g, j = 1$
- while $i > 0$ and $j \le n/g$:
  - if $-c \in A_{i,j}$: return “solution found”
  - if $\min(A_i) + \max(A_j) > -c$: $i := i - 1$
  - otherwise: $j := j + 1$
- return “no solution”



Small Decision Tree

given a set A of n integers, are there $a, b, c \in A$ such that $a + b + c = 0$?

sort A in increasing order — $O(n \log n)$ comparisons

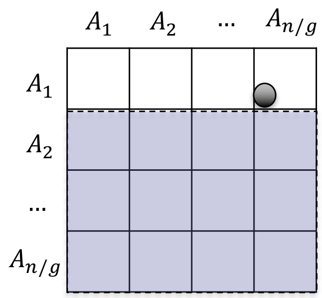
partition A into n/g groups: $A_1, \dots, A_{n/g}$
(all elements of A_i are smaller than all elements of A_{i+1})

sort $D := \bigcup_{i=1}^{n/g} A_i - A_i = \{a - b \mid \exists i: a, b \in A_i\}$ — $O(|D| \log |D|) = O(ng \log(ng))$ comparisons

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!

for each $c \in A$: *check whether there are $a, b \in A$ s.t. $a + b + c = 0$*

- initialize $i = n/g, j = 1$
- while $i > 0$ and $j \le n/g$:
  - if $-c \in A_{i,j}$: return “solution found”
  - if $\min(A_i) + \max(A_j) > -c$: $i := i - 1$
  - otherwise: $j := j + 1$
- return “no solution”



Small Decision Tree

given a set A of n integers, are there $a, b, c \in A$ such that $a + b + c = 0$?

sort A in increasing order — $O(n \log n)$ comparisons

partition A into n/g groups: $A_1, \dots, A_{n/g}$
(all elements of A_i are smaller than all elements of A_{i+1})

sort $D := \bigcup_{i=1}^{n/g} A_i - A_i = \{a - b \mid \exists i: a, b \in A_i\}$ — $O(|D| \log |D|) = O(ng \log(ng))$ comparisons

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!

for each $c \in A$: *check whether there are $a, b \in A$ s.t. $a + b + c = 0$* — n iterations

- initialize $i = n/g, j = 1$
- while $i > 0$ and $j \le n/g$: — $O(n/g)$ iterations
  - if $-c \in A_{i,j}$: return “solution found” — $O(\log(g^2)) = O(\log n)$ comparisons using binary search
  - if $\min(A_i) + \max(A_j) > -c$: $i := i - 1$
  - otherwise: $j := j + 1$
- return “no solution”

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

Decision Tree Complexity

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

Thm: Without bit-tricks, 3SUM is in time $O(n^2 \cdot \frac{\text{poly} \log \log n}{\sqrt{\log n}})$

Outline

1) algorithm for small universe

2) quadratic algorithm

3) small decision tree

4) logfactor improvement

5) some 3SUM-hardness results

Converting Decision Tree to Algorithm

given a set A of n integers, are there $a, b, c \in A$ such that $a + b + c = 0$?

sort A in increasing order — $O(n \log n)$ comparisons and time

partition A into n/g groups: $A_1, \dots, A_{n/g}$

(all elements of A_i are smaller than all elements of A_{i+1})

sort $D \coloneqq \bigcup_{i=1}^{n/g} A_i - A_i = \{a - b \mid \exists i: a, b \in A_i\}$ — $O(|D| \log |D|) = O(ng \log(ng))$ comparisons and time

for all i, i': sort $A_{i,i'} \coloneqq A_i + A_{i'} = \{a + b \mid a \in A_i, b \in A_{i'}\}$ — no comparisons! $O((n/g)^2 \cdot g^2 \log(g^2))$ time

for each $c \in A$: *check whether there are $a, b \in A$ s.t. $a + b + c = 0$* — n iterations

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

while $i > 0$ and $j \leq n/g$: — $O(n/g)$ iterations

if $-c \in A_{i,j}$: return "solution found" — $O(\log(g^2)) = O(\log n)$ comparisons and time using binary search

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

otherwise: $j \coloneqq j + 1$

return "no solution"

in total: $O(n^2 \log(g^2))$ time ☹

Converting Decision Tree to Algorithm

for all i, i': sort $A_{i,i'} \coloneqq A_i + A_{i'} = \{a + b \mid a \in A_i, b \in A_{i'}\}$ write $A_i = \{a_{i,1}, \dots, a_{i,g}\}$

implement this step faster!

simplification:

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

replace A_i by $\{a_{i,j} \cdot (2g)^2 + j \mid 1 \leq j \leq g\}$

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

then 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 the new $A_i + A_{i'}$ we can recover the old $A_i + A_{i'}$

Converting Decision Tree to Algorithm

for all i, i': sort $A_{i,i'} \coloneqq A_i + A_{i'} = \{a + b \mid a \in A_i, b \in A_{i'}\}$ write $A_i = \{a_{i,1}, \dots, a_{i,g}\}$

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

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

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

this is the correct sorted ordering of $A_{i,i'}$ if and only if:

$$a_{i,\pi_k} + a_{i',\sigma_k} < a_{i,\pi_{k+1}} + a_{i',\sigma_{k+1}} \qquad \text{for all } 1 \leq k < g^2$$

by Fredman's trick, this is equivalent to:

$$a_{i',\sigma_k} - a_{i',\sigma_{k+1}} < a_{i,\pi_{k+1}} - a_{i,\pi_k} \qquad \text{for all } 1 \leq k < g^2$$

construct vectors: $(a_{i',\sigma_k} - a_{i',\sigma_{k+1}})_{1 \leq k < g^2}$

$(a_{i,\pi_{k+1}} - a_{i,\pi_k})_{1 \leq k < g^2}$

we say that vector x **dominates** vector y if $x_i > y_i$ for all i

Dominance Reporting

Dominance Reporting problem:

given sets A, B of (integer-valued) vectors in $\mathbb{R}^d$, $|A| + |B| = m$

report all pairs $a \in A, b \in B$ where b dominates a

Thm: Dominance Reporting is in time $O\big(m\,(\log m)^d + \text{outputsize}\big)$

max planck institut informatik

Converting Decision Tree to Algorithm

for all i, i': sort $A_{i,i'} \coloneqq A_i + A_{i'} = \{a + b \mid a \in A_i, b \in A_{i'}\}$ write $A_i = \{a_{i,1}, \dots, a_{i,g}\}$

for each permutation $P = \big((\pi_1, \sigma_1), (\pi_2, \sigma_2), \dots, (\pi_{g^2}, \sigma_{g^2})\big)$ of $\{1, .., g\} \times \{1, \dots, g\}$:

construct sets: $A = \{\ (a_{i',\sigma_k} - a_{i',\sigma_{k+1}})_{1 \le k < g^2} \mid 1 \le i' \le n/g\ \}$

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

solve Dominance Reporting on A, B — time $O(m\,(\log m)^d + \text{outputsize})$

for each reported pair (i, i'):

the sorted ordering of $A_{i,i'}$ is given by P:

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

time for sorting all $A_{i,i'}$: $O\Big((g^2)! \cdot (n/g)(\log n/g)^{g^2} + (n/g)^2\Big)$

- $(g^2)!$ iterations
- one vector for each of n/g groups
- vectors have dimension g^2
- **total** outputsize $(n/g)^2$

max planck institut informatik

Converting Decision Tree to Algorithm

for all i, i': sort $A_{i,i'} \coloneqq A_i + A_{i'} = \{a + b \mid a \in A_i, b \in A_{i'}\}$ write $A_i = \{a_{i,1}, \dots, a_{i,g}\}$

for each permutation $P = \big((\pi_1, \sigma_1), (\pi_2, \sigma_2), \dots, (\pi_{g^2}, \sigma_{g^2})\big)$ of $\{1, .., g\} \times \{1, \dots, g\}$:

construct sets: $A = \{\ (a_{i',\sigma_k} - a_{i',\sigma_{k+1}})_{1 \le k < g^2} \mid 1 \le i' \le n/g\ \}$

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

solve Dominance Reporting on A, B — time $O(m\,(\log m)^d + \text{outputsize})$

for each reported pair (i, i'):

the sorted ordering of $A_{i,i'}$ is given by P:

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

time for sorting all $A_{i,i'}$: $O\Big((g^2)! \cdot (n/g)(\log n/g)^{g^2} + (n/g)^2\Big) \quad = O\big((n/g)^2\big)$

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

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

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

max planck institut informatik

Converting Decision Tree to Algorithm

given a set A of n integers, are there $a, b, c \in A$ such that $a + b + c = 0$?

sort A in increasing order — $O(n \log n)$ comparisons and time

partition A into n/g groups: $A_1, \dots, A_{n/g}$

(all elements of A_i are smaller than all elements of A_{i+1})

sort $D \coloneqq \bigcup_{i=1}^{n/g} A_i - A_i = \{a - b \mid \exists i: a, b \in A_i\}$ — $O(|D| \log|D|) = O(ng \log(ng))$ comparisons and time

for all i, i': sort $A_{i,i'} \coloneqq A_i + A_{i'} = \{a + b \mid a \in A_i, b \in A_{i'}\}$ — no comparisons! $O((n/g)^2 \cdot$ ~~$g^2 \log(g^2)$~~$)$ time

for each $c \in A$: *check whether there are $a, b \in A$ s.t. $a + b + c = 0$* — n iterations

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

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

if $-c \in A_{i,j}$: return “solution found” — $O(\log(g^2)) = O(\log n)$ comparisons and time using binary search

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

otherwise: $j \coloneqq j + 1$

return “no solution”

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

for $g \coloneqq 0.1 \cdot \sqrt{\log n / \log\log n}$: $O\big(n^2 \frac{\text{poly}\log\log(n)}{\sqrt{\log n}}\big)$ time

max planck institut informatik

Decision Tree Complexity

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

Thm: Without bit-tricks, 3SUM is in time $O(n^2 \cdot \frac{\text{poly} \log \log n}{\sqrt{\log n}})$

Dominance Reporting

Dominance Reporting problem:

given sets A, B of (integer-valued) vectors in $\mathbb{R}^d$, $|A| + |B| = m$

report all pairs $a \in A, b \in B$ where b dominates a

Thm: Dominance Reporting is in time $O\big(m\,(\log m)^d + \text{outputsize}\big)$

deciding whether there is a dominating pair (a, b) is OV-hard

so we do not expect an $O(\text{poly}(d)\, m^{2-\varepsilon})$ algorithm

OV is in time $O(2^d m)$

the theorem „generalizes“ this OV-algorithm to Dominance Reporting

Dominance Reporting

Dominance Reporting problem:

given sets A, B of (integer-valued) vectors in $\mathbb{R}^d$, $|A| + |B| = m$

report all pairs $a \in A, b \in B$ where b dominates a

Thm: Dominance Reporting is in time $O\big(m\,(\log m)^d + \text{outputsize}\big)$

assume all coordinates to be different

if $d = 0$: report all pairs $A \times B$ $\quad T_d(m) \le 2T_d(m/2) + T_{d-1}(m) + m$

otherwise:

- find median x of d-th coordinates of all points in $A \cup B$ - time $O(m)$
- $A_S := \{a \in A \mid a_d < x\}$ and $A_L := A \setminus A_S$
- $B_S := \{b \in B \mid b_d < x\}$ and $B_L := B \setminus B_S$
- recursively solve $(A_L, B_L), (A_S, B_S)$, and (A_S, B_L)

↑ remove d-th coordinates!

Dominance Reporting

$T_d(m) \le 2T_d(m/2) + T_{d-1}(m) + m$

Excluding cost of output: $T_0(m) = T_d(1) = 0$

Inductively prove that: $T_d(m) \le m\,(\log 2m)^d - m$

$$T_d(m) \le 2\left(\frac{m}{2}\,(\log m)^d - \frac{m}{2}\right) + \big(m\,(\log 2m)^{d-1} - m\big) + m$$
$$= m\,((\log 2m) - 1)^d + m\,(\log 2m)^{d-1} - m$$
$$= m\,(\log 2m)^d (1 - 1/\log 2m)^d + m\,(\log 2m)^{d-1} - m$$
$$\le m\,(\log 2m)^d (1 - 1/\log 2m) + m\,(\log 2m)^{d-1} - m$$
$$= m\,(\log 2m)^d - m$$

Dominance Reporting

Dominance Reporting problem:

given sets A, B of (integer-valued) vectors in $\mathbb{R}^d$, $|A| + |B| = m$

report all pairs $a \in A, b \in B$ where b dominates a

Thm: Dominance Reporting is in time $O\big(m\,(\log m)^d + \text{outputsize}\big)$

this finishes the proof of:

Thm: Without bit-tricks, 3SUM is in time $O(n^2 \cdot \frac{\text{poly}\log\log n}{\sqrt{\log n}})$

max planck institut informatik