Fast “Space-Efficient” Subset Sum

Nikhil Bansal (CWI and TU/e)

Shashwat Garg, Jesper Nederlof (TU/e)

Nikhil Vyas (MIT)

STOC 2017

Subset Sum Problem

Input: integers $w_1, \ldots, w_n$ target = t
Goal: is there a subset $X \subseteq [n]$ with $\sum_{i \in X} w_i = t$?

Notation: w = $(w_1, \ldots, w_n)$ is there x ∈ $\{0,1\}^n$ s.t. $w \cdot x = t$

The `classic' results:

- Trivial enumeration: $\mathrm{O}^*(2^{\mathrm{n}})$ time, $\mathrm{O}^*(1)$ space

O*( ) hides poly(n) factors, dependence on bit lengths

- Dynamic Programming: O*(nt) time, O*(nt) space

T[i,v]= 1 iff some subset of first i elements sums to v
T[i,v] = max (T[i-1,v] T[i-1,v-w_i])

Can assume $t \approx 2^n$

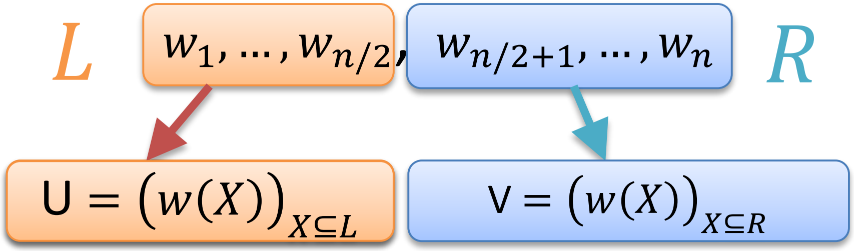
Meet in the Middle

Horowitz-Sahni'72: $O^*(2^{n/2})$ time, $O^*(2^{n/2})$ space

Let L = $(w_1, \dots, w_{n/2})$ R = $(w_{n/2+1}, \dots, w_n)$

Compute all possible $2^{n/2}$ sums:

U and V.



L $w_1, \dots, w_{n/2},$ $w_{n/2+1}, \dots, w_n$ R

U = $\big(w(X)\big)_{X \subseteq L}$ V = $\big(w(X)\big)_{X \subseteq R}$

For every u ∈ U, check if t-u ∈ V ?

Sort V + binary search

Schroeppel-Shamir'81: $O^*(2^{n/2})$ time, $O^*(2^{n/4})$ space

State of the art

Q1: Any $(2^{n(1/2-\epsilon)})$ time algorithm ?

(something that beats meet in the middle)

Q2: Any $(2^{n(1-\epsilon)})$ time, poly(n) space algorithm?

(something that beats enumeration)

Random instances (w_i uniformly chosen in $[2^n]$)

$2^{0.3113}$ n [Howgrave-Graham, Joux'10]

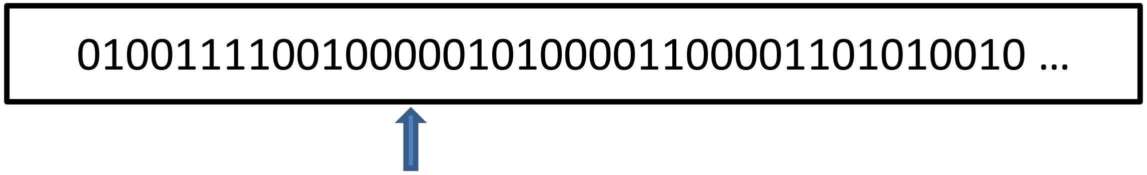
$2^{0.72}$ n, poly space [Becker, Coron, Joux'11] (claimed)

Our result

Thm: There is a randomized algorithm for subset sum that uses

$O^*(2^{0.86\ n})$ time, and O*(1) space

Assuming read-only random access to exponentially many random bits



010011110010000010100001100001101010010 …

Read-only tape

Random access

Need to implement random hash functions

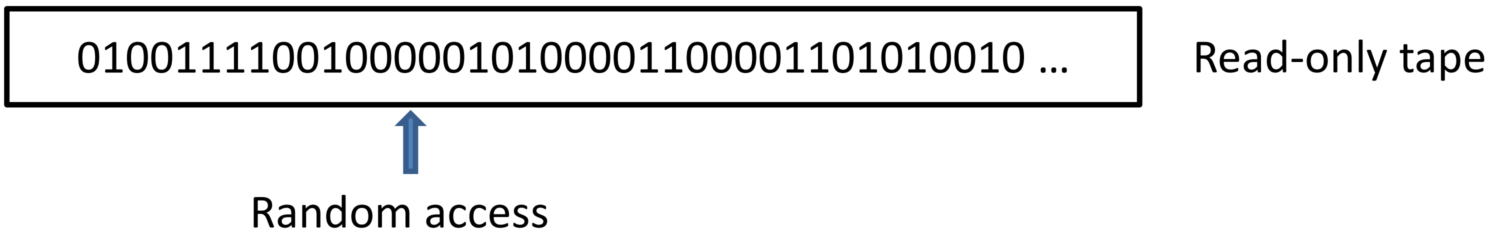
Can also assume suitable PRG (e.g. for space bounded computation)

Our result

Thm: There is a randomized algorithm for subset sum that uses

$O^*(2^{0.86\ n})$ time, and O*(1) space

Assuming read-only random access to exponentially many random bits



010011110010000010100001100001101010010 ...

Read-only tape

Random access

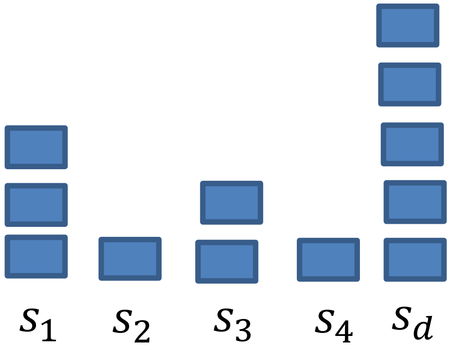
Result also extends to other problems

Knapsack, k-sum, d-dim. binary integer programs, ...

Idea

Consider $w(2^{[n]})$ = $\{w \cdot x : \text{x} \in \{0,1\}^n\}$

i.e. all 2^n subset sums of $w = (w_1, \ldots, w_n)$



Histogram of sums

Let d = $|w(2^{[n]})|$ i.e. # distinct possible sums

Case 1: If d < $2^{0.86n}$ (few buckets)

a) Work mod O(d), which makes t = O(d)

b) O*(t) time DP, but in O*(1) space [Nederlof-Lokshtanov'10]

Case 2: If d > $2^{0.86n}$ (many buckets)

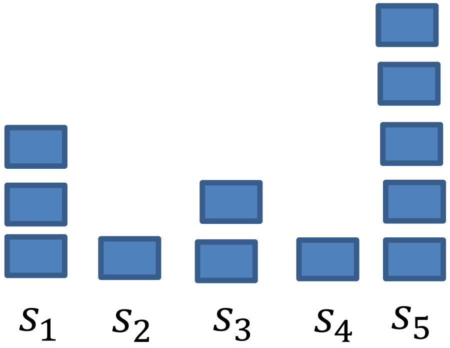
a) Basic additive combinatorics

b) Floyd's cycle finding for list disjointness in O*(1) space

Case 1: Few buckets Case

Let d = |w($2^{[n]}$)| i.e. # distinct possible sums

d < $2^{0.86n}$



Pick a random number p between [10d,20d]

Solve subset sum mod p.

Pr[false positive] ≤ 1/2

i.e. Pr[∃ t' ∈ w($2^{[n]}$) : t' ≡ t (mod p)] ≤ ½

As $t = O(\mathrm{d})$, can use the O(nt) time dynamic program

(but space is large)

O(1) space DP

Thm (Lokshtanov-Nederlof’10): Can solve subset sum DP in O(n^3t) time, O(n^3) space, via algebraization

Generalizes to a wide class of arithmetic circuits

Subset Sum

Proof: Subset sum = coefficient of x^t is 0 or not in

f(x) = $(1+x^{w_1})(1+x^{w_2})\cdots(1+x^{w_n})$

Given f(x) implicitly (via evaluation oracle), find coefficient of x^t.

Easy Fact: For any degree d polynomial f(x)

Coefficient of x^t = 1/p $\sum_{i=0}^{p-1}$ $f(\omega_p^i)\,\omega_p^{-it}$

ω_p= p-th root of unity, p>d

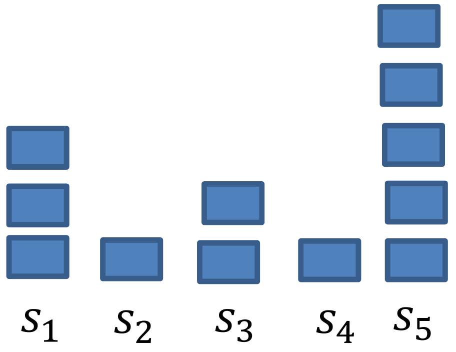
Need O*(1) space: Can evaluate $f(\omega_p^i)$ sequentially.

O(n) bits of numerical precision suffice.

Many buckets Case (d > $2^{0.86n}$)

d = |w($2^{[n]}$)| i.e. # distinct sums

Imagine $d \approx 2^n$



Thm (easy version): $\mathrm{d} \cdot b_{max} \leq 3^n$

b_{max}: max bucket size

Apriori, $\mathrm{d} \cdot b_{max}$ can be almost 4^n (so, subset sums are "smoothly" distributed)

Intuition: Suppose $w \cdot x = s$ for several $x \in \{0,1\}^n$

Then lots of "dependencies" (i.e. $y \in \{-1,0,1\}^n$ with $w \cdot y = 0$)

Should give collisions in many other buckets

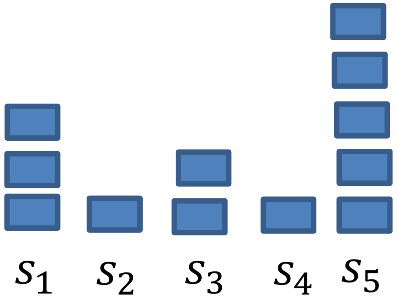
Smoothness ($d \cdot b_{max} \leq 3^n$)

Thm: Let $A \subset \{0,1\}^n$ and $B \subset \{0,1\}^n$, s.t. for all $a \in A, b \in B$, a+b (over Z^n) uniquely determines a and b, then |A||B| $\leq 3^n$.

Proof: Trivial as |A||B|=|A+B| $\leq 3^n$.

Subset Sum: A = one vector from each bucket

B = vectors in max bucket.



Clearly, |A| = d |B| = b_{max}

Say a+b = a'+b'. Then $w \cdot (a + b) = w \cdot (a' + b')$

$w \cdot a = w \cdot a'$ (as $w \cdot b = w \cdot b'$)

Contradicts that a and a' lie in different buckets

UDCP: Unique decodable code pairs

We use a refined version, tailored to ℓ_2 norm of bucket frequencies

So Far

If d < $2^{0.86\,n}$ (easy by hashing)

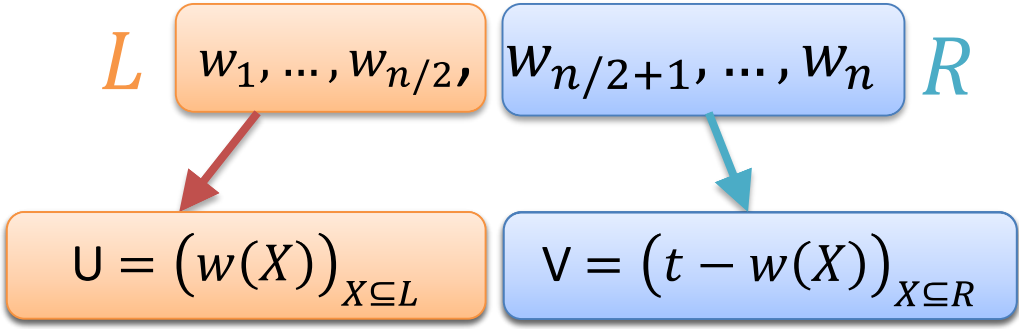
If d $>$ $2^{0.86\,n}$, $b_{\max}$ < $3^n/d$ < $2^{0.73\,n}$ (smoothness)

Final piece: Use smoothness to give a faster algorithm (using bounded space)

List-disjointness view of subset sum

Recall meet in the middle

Lists U and V of size $2^{n/2}$



L $w_1, \dots, w_{n/2},$ $w_{n/2+1}, \dots, w_n$ R

$\mathrm{U} = \left(w(X)\right)_{X \subseteq L}$

$\mathrm{V} = \left(t - w(X)\right)_{X \subseteq R}$

List disjointness (LD): Is there a common entry $U_i = V_j$

Bounded space: We cannot store U and V.

But can generate j-th element in U or V for $j \in [2^{n/2}]$

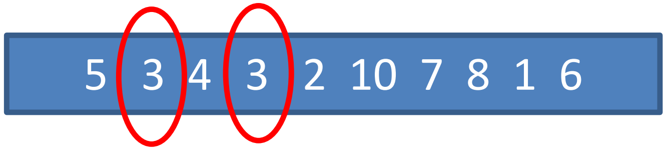
Given two lists U,V of size m=$2^{n/2}$ with random read access.

Is there a common element among U and V?

Element Distinctness

Element distinctness: Given a list L of m integers, is there a common (repeated) element?

Yes



5 3 4 3 2 10 7 8 1 6

Solution: Sort, O(m log m) time.

Space = m

What if O(1) space?

(L given on a read only random access tape)

Check all pairs.

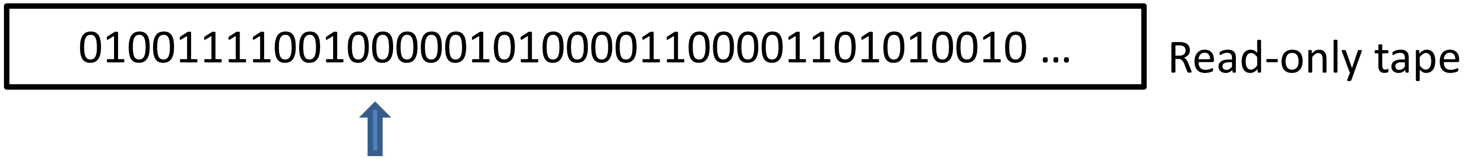
Check a random pair u and v.

All take $\Omega(m^2)$

Element Distinctness

Thm [Beame, Clifford, Machmouchi'13]: Can solve element distinctness in $O(m^{3/2})$ time and O(1) space.

Assuming read-only random access tape with random bits.

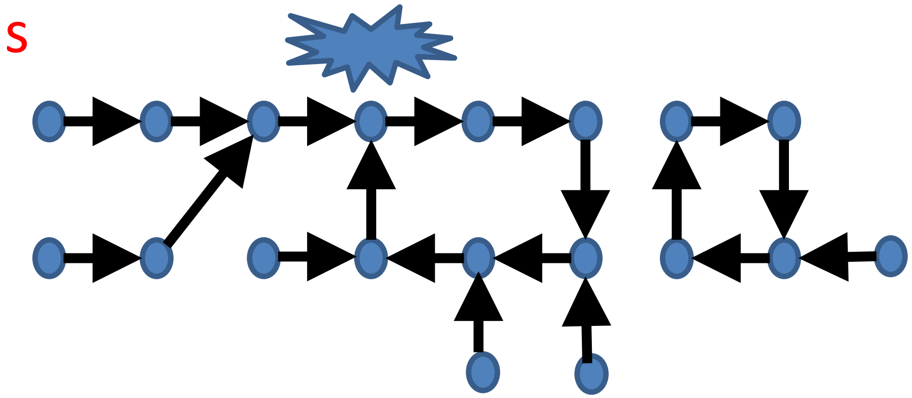


Main tool: Floyd's cycle finding

Widely used in cryptography (aka Pollard Rho)

Floyd’s Cycle Finding

Given an (implicit) outdegree 1 digraph. Find a collision.



Floyd’s algorithm: Given s, can find collision using 2 pointers, in time O(stem + cycle)

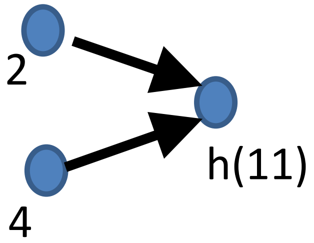
Using Cycle Finding

Element distinctness: List L[1],…,L[m]

27 11 6 11 9 m=5

Hash function h: L -> [m]

digraph: Vertices [m], Edges: i -> h(L[i])



Proof: [BCM'13]

h random, can find collision in time $n^{1/2}$ (birthday paradox)

Repeat n times (fake collisions due to hashing).

List-disjointness: Can get fake collisions or collisions within a list.

Use smoothness to control the damage.

$\mathrm{U} = \big(w(X)\big)_{X\subseteq L}$

$\mathrm{V} = \big(t - w(X)\big)_{X\subseteq R}$

Natural Open Questions

- How strong is random bits assumption exactly?
  - Weaker than the existence of sufficiently strong PRG's
  - Still don't know the exact (low-space) complexity of ED

- Solve Subset Sum in time $O^*\left(2^{(0.5-\epsilon)n}\right)$
  - progress on many cases (Austrin-Kaski-Koivisto-Nederlof'16)