15-853:Algorithms in the Real World

Nearest Neighbors
- -Callahan-Kosaraju
- -Use in all nearest neighbors
- -Use in N-body codes

Callahan-Kosaraju

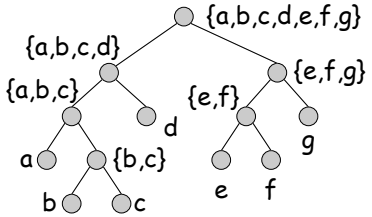
Well separated pair decompositions
- A decomposition of points in d-dimensional space

Applications
- N-body codes (calculate interaction forces among n bodys)
- K-nearest-neighbors O(n log n) time

Similar to k-d trees (e.g. quad-trees) but better theoretical properties

Tree decompositions

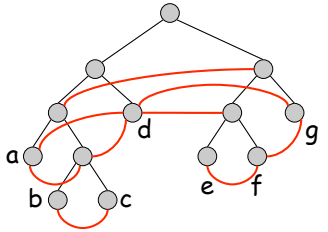
A spatial decomposition of points



A "realization"

A single path between any two leaves consisting of tree edges up, an interaction edge across, and tree edges down.

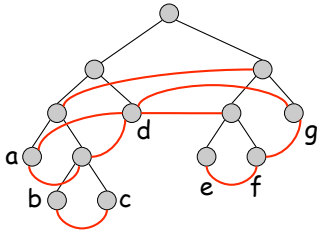
interaction edge



A "well-separated realization"

A realization such that the endpoints of each interaction edge is "well separated"

Goal: show that the number of interaction edges is O(n)



Overall approach

Build tree decomposition: O(n log n) time

Build well-separated realization: O(n) time

Depth of tree = O(n) worst case, but not in practice

We can bound number of interaction edges to O(n)

- For both n-body and nearest-neighbors we only need to look at the interaction edges

Callahan Kosaraju Outline

Some definitions

Building the tree

Generating well separated realization

Bounding the size of the realization

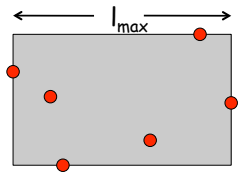
Using it for nearest neighbors

Some Definitions

Bounding Rectangle R(P)

Smallest rectangle that contains a set of points P

l_{max} : maximum length of a rectangle

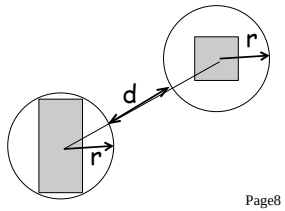


Well Separated:

r = smallest radius that can contain either rectangle

s = separation constant

d > s * r



More Definitions

Interaction Product

$A \otimes B = \{\{p,p'\} : p \in A, p' \in B, p \neq p'\}$

A **Realization** of $A \otimes B$

Is a set $\{\{A_1,B_1\}, \{A_2,A_3\}, \ldots, \{A_k,B_k\}\}$

such that

1. $A_i \subseteq A$, $B_i \subseteq B$ $\quad i = 1 \ldots k$
2. $A_i \cap B_i = \varnothing$
3. $(A_i \otimes B_i) \cap (A_j \otimes B_j) = \varnothing \quad (i \neq j)$
4. $A \otimes B = \cup_{i=1}^{k} A_i \otimes B_i$

This formalize the "cross edges"

A well-separated realization

$\{\{A_1,B_1\}, \{A_2,A_3\}, \ldots, \{A_k,B_k\}\}$

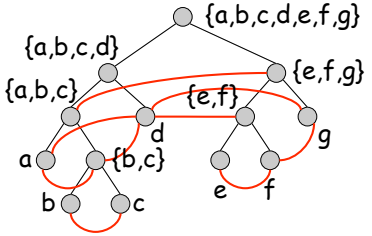
such that $R(A_i)$ and $R(B_i)$ are well separated

A well-separated pair decomposition =

Tree decomposition of P

+ well-separated realization of $P \otimes P$ where the subsets are the nodes of the tree

A well-separated pair decomposition



P = {a,b,c,d,e,f,g}

Realization of $P \otimes P$ =

{{{a,b,c},{e,f,g}}, {{d},{e,f}}, {{d},{b,c}}, {{a},{b,c}}, {{a},{d}}, {{b},{c}}, {{d},{g}}, {{e},{f}}}

Algorithm: Build Tree

Function Tree(P)

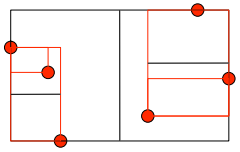
if |P| = 1 then **return** leaf(P)

else

d_{max} = dimension of l_{max}

P_1, P_2 = split P along d_{max} at midpoint

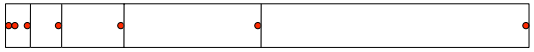
Return Node(Tree(P_1),Tree(P_2), l_{max})



Runtime: naive

Naively:

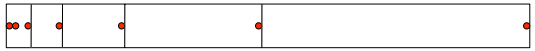
Each cut could remove just one point



$T(n) = T(n-1) + O(n) = O(n^2)$

This is no good!!

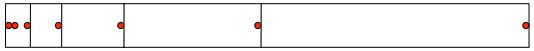
Runtime: better



1. Keep points in linked list sorted by each dimension
2. In selected dimension come in from both sides until cut is found
3. Remove cut elements and put aside
4. Repeat making cuts until size of largest subset is less than 2/3 n
5. Create subsets and make recursive calls

$$T(n) = \sum_{i=1}^{k} T(n_i) + O(n)$$

Runtime: better



$$T(n) = \sum_{i=1}^{k} T(n_i) + O(n)$$

$$\text{such that}: \sum_{i=1}^{k} n_i = n$$

$$\text{and}: \max_{i=1}^{k}(n_i) < \frac{2}{3}n$$

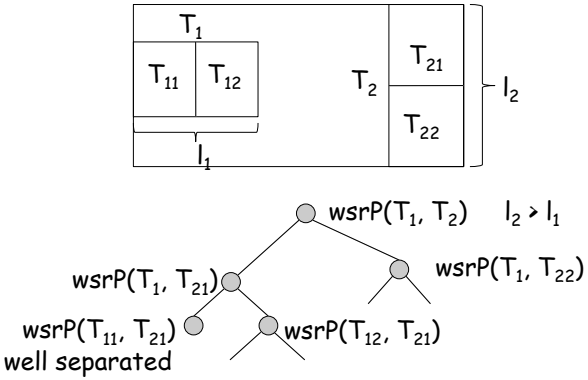
$$T(n) = O(n \lg n)$$

Algorithm: Generating the Realization

```
function wsr(T)
if leaf(T) return ∅
else return wsr(left(T)) ∪ wsr(right(T))
            ∪ wsrP(left(T),right(T)

function wsrP(T1, T2)
if wellSep(T1,T2) return {(T1,T2)}
else if lmax(T1) > lmax(T2) then
  return wsrP(left(T1), T2) ∪ wsrP(right(T1), T2)
else
  return wsrP(T1, left(T2)) ∪ wsrP(T1, right(T2))
```

WSR



Bounding Interactions

Just an intuitive outline:

- Can show that tree nodes do not get too thin
- Can bound # of non-overlapping rectangles that can touch a cube of fixed size
- Can bound number of interaction per tree node

Total calls to wsrP is bounded by

$$2n\left(2\left(s\sqrt{d}+2\sqrt{d}+1\right)+2\right)^{d}=O(n)$$

This bounds both the time for WSR and the number of interaction edges created.

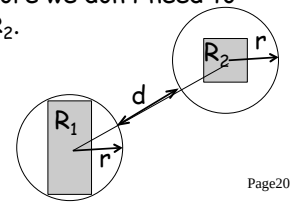
Summary so far

O(n log n) time to build tree
O(n) time to calculate WS Pair Decomposition
O(n) edges in decomposition

Finding everyone's nearest neighbor

Build well-separated pair decomposition with s = 2.

- Recall that d > sr = 2r to be well separated
- The furthest any pair of points can be to each other within one of the rectangles is 2r
- Therefore if d > 2r then for a point in R_1 there must be another point in R_1 that is closer than any point in R_2. Therefore we don't need to consider any points in R_2.



Finding everyone's nearest neighbor

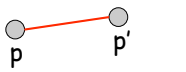
Now consider a point p.

It interacts with all other points p' through an interaction edge that goes from:

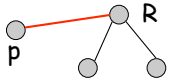
1. p to p'
   (check these distances directly)
2. p to an ancestor R of p'
   (check distance to all descendants of R)
3. an ancestor of p to p' or ancestor of p'
   (p' cannot be closest node)

Step 2 might not be efficient, but efficient in practice and can be made efficient in theory

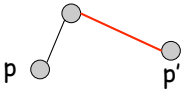
Again: in pictures



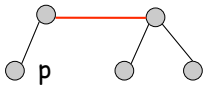
calculate d(p,p')



for p' in R, calculate d(p,p')



ignore for p



ignore

Take minimum of all distances calculated.

The N-body problem

Calculate the forces among n "bodys". Naïve method requires considering all pairs and takes $O(n^2)$ time.

Using Kallahan-Kosaraju can get approximate answer in O(n) time plus the time to build the tree.

Used in astronomy to simulate the motion of starts and other mass

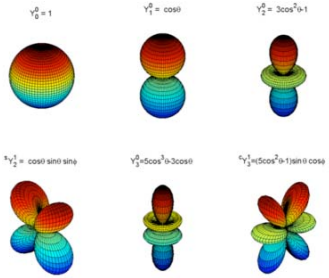
Used in biology to simulate protein folding

Used in engineering to simulate PDEs (can be better than Finite Element Meshes for certain problems)

Used in machine learning to calculate certain Kernels

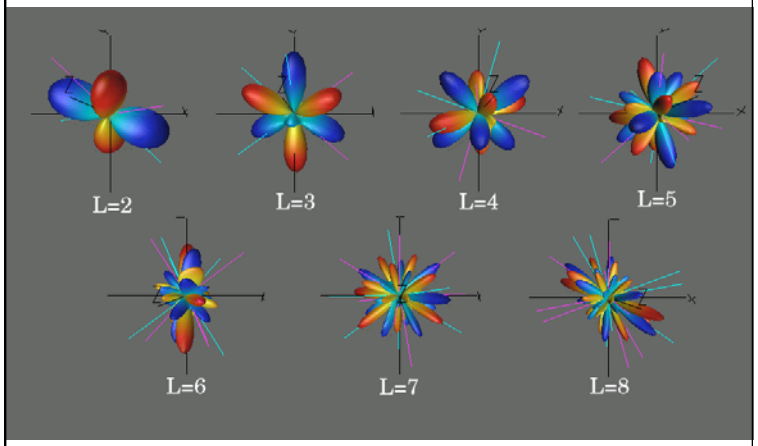
The N-body problem

Can approximate the force/potential due to a set of points by a multipole expansion truncated to a fixed number of terms (sort of like a taylor series).



The spherical harmonics

Potential due to Y_l term goes off as $1/r^{l+1}$ so far away the low terms dominate.



The N-body problem

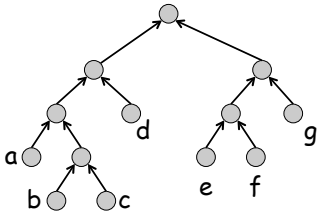
If a set of points in well-separated from p, then can use the approximation instead of all forces.

Need "inverse" expansion to pass potential down from parents to children.

The N-body problem

If a set of points in well-separated from p, then can use the approximation instead of all forces.

Need "inverse" expansion to pass potential down from parents to children.

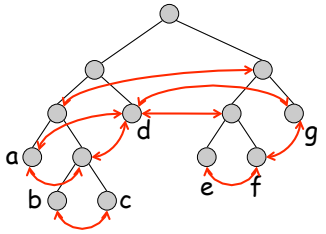


Translate and add "multipole" terms going up the tree. They add linearly.

The N-body problem

If a set of points in well-separated from p, then can use the approximation instead of all forces

Need "inverse" expansion to pass potential down from parents to children.

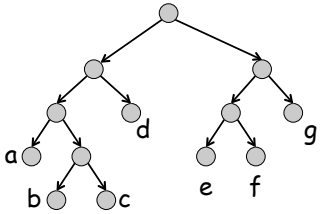


Invert expansions across the interaction edges.

The N-body problem

If a set of points in well-separated from p, then can use the approximation instead of all forces

Need "inverse" expansion to pass potential down from parents to children.



Copy add and translate the inverse expansions down the tree. Calculate approximate total force at the leaves.

The N-body problem

If a set of points in well-separated from p, then can use the approximation instead of all forces.

Need "inverse" expansion to pass potential down from parents to children.

Total time is:

- O(n) going up the tree
- O(n) inverting across interaction edges
- O(n) going down the tree

The constant in the big-O and the accuracy depend on the number of terms used. More terms is more costly but more accurate.

The N-body problem