DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Università degli Studi di Padova – Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica

Relazione per la prova finale

«STRUTTURE DATI PER LA GESTIONE DI DATI SPAZIALI»

Relatore: Prof. Pietracaprina Andrea

Laureando: Mattia Favretto
Matricola n.: 2101833

Padova, 21/07/2026

OBIETTIVO

Studio e analisi di Kd-Tree, Range Tree, Quadtree, R-Tree ed R*-Tree.

Fonti principali:
1. «M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. Computational Geometry: Algorithms and Applications.»
2. «Antonin Guttman. R-trees: A dynamic index structure for spatial searching.»
3. «Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: An efficient and robust access method for points and rectangles.»

NOTAZIONE

- n Numero di punti dell'insieme considerato
- d Numero di dimensioni
- k Numero di punti riportati in una query
- v Nodo generico dell'albero
- p Punto generico nello spazio
- P Insieme di punti

INTRODUZIONE

Kd-tree

Range tree

Quadtree

Conservano punti in R^d.

R-Tree

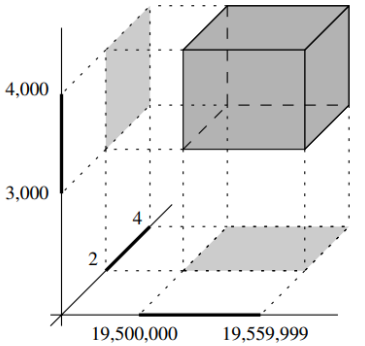
R*-Tree

Conservano oggetti con reale estensione spaziale in R^d oppure punti in R^d.

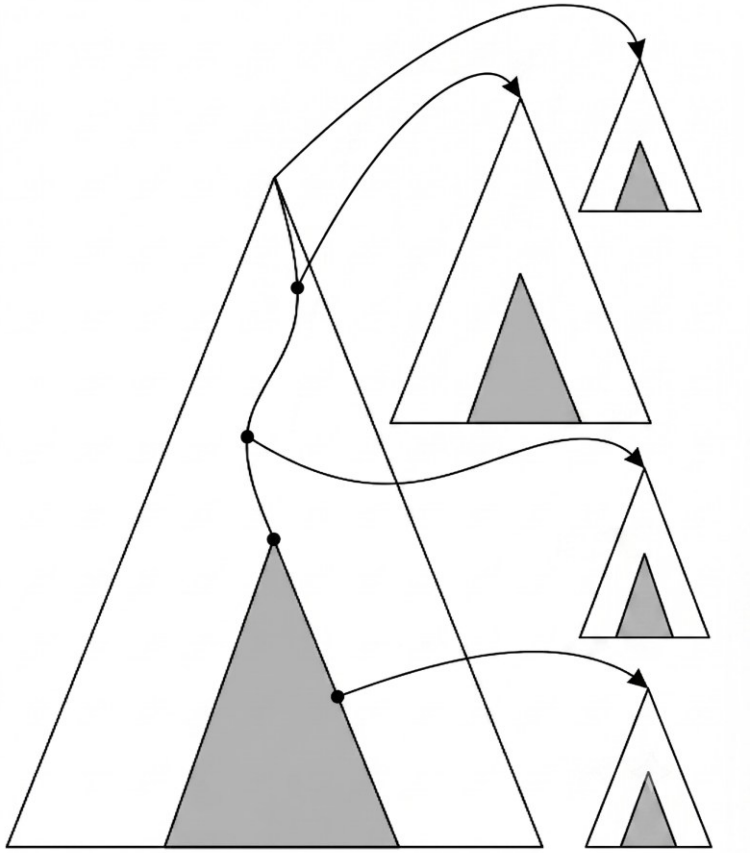
Orthogonal range query

Input: insieme P di punti in d-dimensioni

Output: punti appartenenti ad un rettangolo d-dimensionale con lati paralleli agli assi



RANGE TREE



Il range tree è una struttura dati per la conservazione di punti in uno spazio R^d. Nella sua versione bidimensionale è costituito da un albero principale costruito sulle coordinate di x, in cui ciascun nodo v è legato ad un albero secondario $\mathrm{J}_{\mathrm{ass}}(v)$ costruito sulle coordinate y di $P(v)$. I punti sono conservati nelle foglie dell'albero secondario.

$P(v)$ è il sottoinsieme canonico, ossia l'insieme dei punti nel sottoalbero radicato in v.

Generalizzabile per contenere punti in R^d, costruendo ricorsivamente strutture associate rispetto ad una coordinata $i \mid 1 \leq i \leq d$.

RANGE TREE

Spazio

$O(n \log^{d-1} n)$

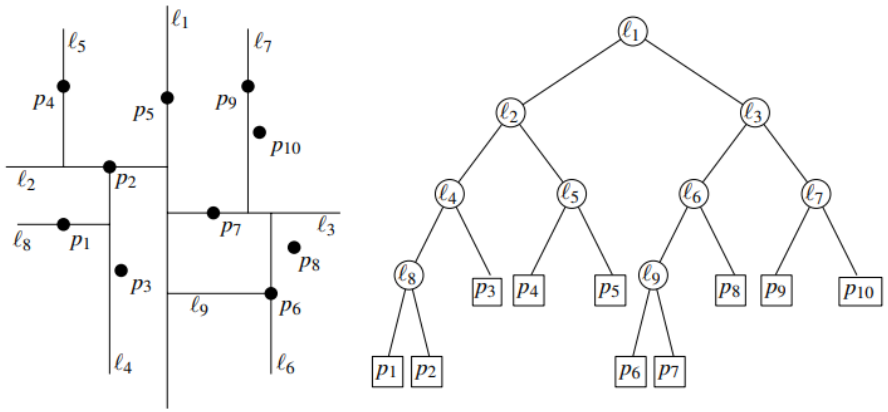
Costruzione

$O(n \log^{d-1} n)$

Range query

$O(\log^{d-1} n + k)$

KD-TREE



Il kd-tree è una struttura costituita da un albero binario nelle cui foglie sono associati i punti di P, mentre i nodi interni sono associati a regioni che definiscono una decomposizione gerarchica dello spazio.

KD-TREE

Spazio

$O(n)$

Costruzione

$O(n \log n)$

Nearest neighbour

Ricerca degli m punti più vicini ad un punto.dato.

$O(\log n)$

Range query

$O\left(n^{1-1/d} + k\right)$

COSTRUZIONE KD-TREE

Algorithm 3: BUILDKDTREE

```
1  if P contains only one point then
2      return a leaf storing this point;
3  else
4      if depth is even then
5          Split P with a vertical line l through the median x-coordinate;
6          P_left ← points to the left or on l;
7          P_right ← points to the right of l;
8      else
9          Split P with a horizontal line l through the median y-coordinate;
10         P_left ← points below or on l;
11         P_right ← points above l;
12     end
13     v_left ← BUILDKDTREE(P_left, depth + 1);
14     v_right ← BUILDKDTREE(P_right, depth + 1);
15     Create node v storing l;
16     Make v_left the left child of v and v_right the right child of v;
17     return v;
18 end
```

$\lceil n/2 \rceil$-esimo valore più piccolo

Input: insieme $P = p_1, p_2, \ldots | p_i \in R^2$ e profondità corrente $depth$

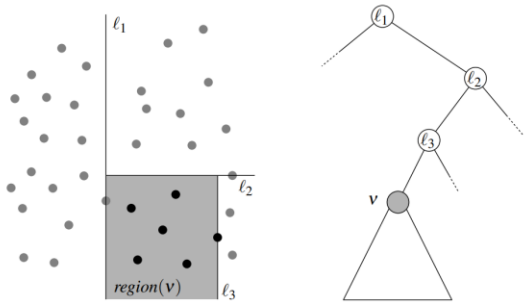
Output: radice di un kd-tree contenente i punti di P

$$T(n) = \begin{cases} O(1) & \text{if } n = 1 \\ O(n) + 2T(\lceil n/2 \rceil) & \text{if } n > 1 \end{cases}$$

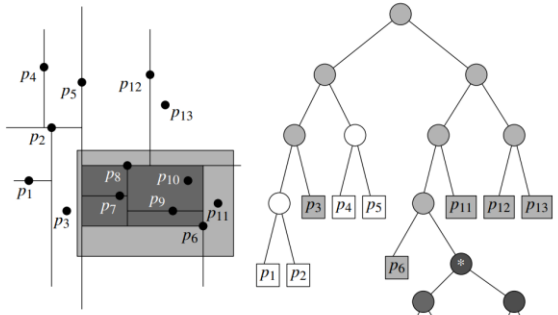
Per ogni livello il costo è di $O(n)$, essendoci $O(\log n)$ livelli il costo totale è $O(n \log n)$.

Generalizzabile per contenere punti in R^d, dividendo i punti dell'insieme ciclicamente rispetto ad una coordinata $i \mid 1 \leq i \leq d$.

RANGE QUERY KD-TREE



Ad ogni nodo v corrisponde una regione spaziale, chiamata $region(v)$.



Esploro completamente una regione se R interseca $region(v)$, se invece la regione è completamente contenuta in R riporto tutti i punti.

$R = [x; x'] \times [y; y']$

RANGE QUERY KD-TREE

Algorithm 4: SEARCHKDTREE

```
1  if v is a leaf then
2      Report the point stored at v if it is in R;
3  else
4      if region(lc(v)) is fully contained in R then
5          REPORTSUBTREE(lc(v));
6      else
7          if region(lc(v)) intersects R then
8              SEARCHKDTREE(lc(v), R);
9          else
10         end
11     end
12     if region(rc(v)) is fully contained in R then
13         REPORTSUBTREE(rc(v));
14     else
15         if region(rc(v)) intersects R then
16             SEARCHKDTREE(rc(v), R);
17         else
18         end
19     end
20 end
```

Riporta le foglie di un sottoalbero in tempo $O(k)$.

Regione del figlio destro di v.

Input: *la radice di un kd-tree e un range* R

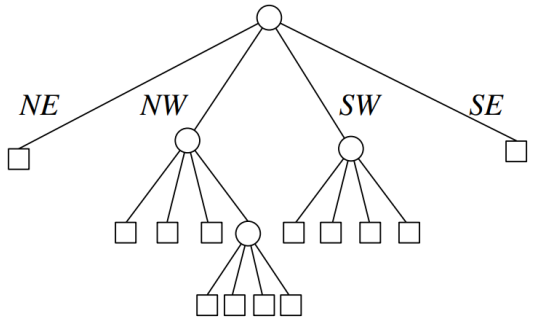
Output: *tutti i punti contenuti nelle foglie, appartenenti al range* R.

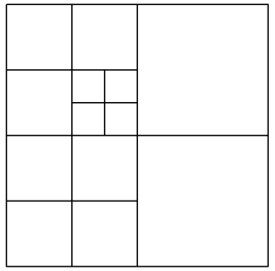
$$Q(n) = \begin{cases} O(1) & \text{if } n = 1 \\ 2 + 2Q(n/4) & \text{if } n > 1 \end{cases}$$

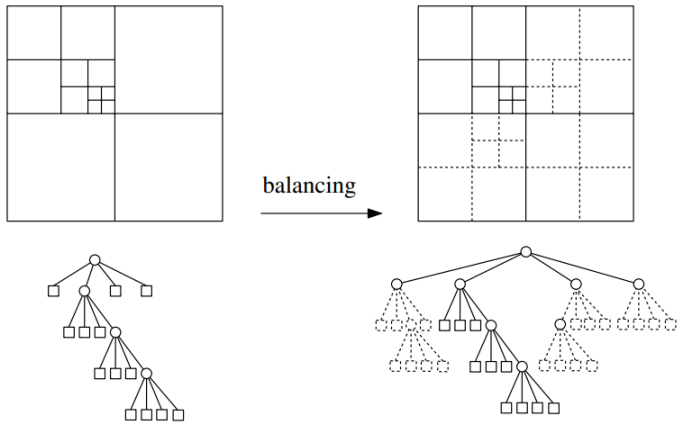
$Q(n)$ rappresenta il numero di regioni intersecate dal rettangolo di query ed equivale a $O(\sqrt{n})$.

Il costo totale è $O(\sqrt{n} + k)$.

QUADTREE







$O(n\ (depth + 1))$

I quadtree sono degli alberi in cui ogni nodo interno ha esattamente quattro figli, a ciascun nodo v è associato un insieme di punti P_v e un quadrato $\sigma(v)$. I nodi foglia contengono i punti di P.

Per il bilanciamento devo assicurare che $max(|depth(\sigma) - depth(\sigma')|) \leq 1$

Il neighbour finding permette la ricerca di nodi con quadrato adiacente al quadrato del nodo dato rispetto ad una direzione (N, S, E, W).

QUADTREE

Spazio

$O(n\ (depth + 1))$

Range query

$O\left(n^{1-1/d} + k\right)$

Costruzione

$O(n\ (depth + 1))$

Neighbour finding

$O(depth + 1)$

R-TREE

L'R-Tree è un albero ad altezza bilanciata.

1. Nelle foglie si trovano puntatori in memoria agli oggetti conservati
$$(I, tuple - identifier)$$
2. Nei nodi interni si trovano puntatori ai figli.
$$((I, child - pointer), \dots.)$$

In ogni nodo si trova un riferimento al rettangolo d-dimensionale che li contiene.
$$I = (I_0, I_1, \dots, I_d)$$

Aspetti fondamentali:

1. Numero di *entris (*numero di figli di un nodo interno).
2. Scelta dell'algoritmo di divisione delle $M + 1$ entries in caso di overflow.

Negli R-Tree si considera solamente il volume dei rettangoli d-dimensionali per la divisione delle entries.

R*-Tree

Negli R*-Tree si considerano:

1. Volume dei rettangoli
2. Overlap tra i rettangoli
3. Perimetro totale

Introduce inoltre il reinserimento forzato in caso di overflow.

CONFRONTI

Kd-tree

- Ricerca degli m punti più vicini rispetto ad un punto
- Efficienza spaziale

Quad tree

- Aggiornamenti frequenti
- Albero non bilanciato

R-tree

- Conservazione di dati spaziali con dimensioni non nulle
- Progettata per operare su disco.

Range tree

- Ottima per il range query in due dimensioni con costo $O(\log n + k)$ ma occupazione spaziale $O\left(\frac{n \log n}{\log \log n}\right)$.

BIBLIOGRAFIA

[1] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The r*-tree: An efficient and robust access method for points and rectangles. In Proceedings of the 1990 ACM SIGMOD international conference on Management of data, pages 322-331, 1990.

[2] Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel. The x-tree: An index structure for high-dimensional data. In Proceedings of the 22th International Conference on Very Large Data Bases, VLDB '96, page 28-39, San Francisco, CA, USA,
1996. Morgan Kaufmann Publishers Inc.

[3] Bernard Chazelle. Filtering search: A new approach to query-answering. In 24° Annual Symposium on Foundations of Computer Science (sfcs 1983), pages 122-132, 1983.

[4] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. Computational Geometry: Algorithms and Applications. Computer Science. Springer Berlin Heidelberg, 2008.

[5] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. ACM Trans. Math. Softw.,3(3):209-226, September 1977.

[6] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In Proceedings of the 1984 ACM SIGMOD international conference on Management of data, pages 47-57, 1984.

[7] D. T. Lee and C. K. Wong. Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees. Acta Inf., 9(1):23-29, March 1977.

[8] Mark H. Overmars. The design of dynamic data structures. In Lecture Notes in Computer Science, 1987.