Maximum Inner-Product Search Using Cone Trees

Guihong Ma, Chuhan Zhang

May 11th 2017

Ram P, Gray A G. Maximum inner-product search using cone trees[C]//Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2012: 931-939.

What is Inner-Product?

- Definition
  - **(Algebraic Definition)** For two arbitrary points $o, q \in \mathbb{R}^d$, the ***inner product*** of o and q is defined as:

$$\langle o, q \rangle = \sum_{i=1}^{d} (o_i * q_i)$$

  - **(Geometric Definition)** If the angle between o and q is $\theta_{o,q}$, the ***inner product*** of o and q can also be defined as:

$$\langle o, q \rangle = \| o \| \cdot \| q \| \cdot cos\theta_{o,q}$$

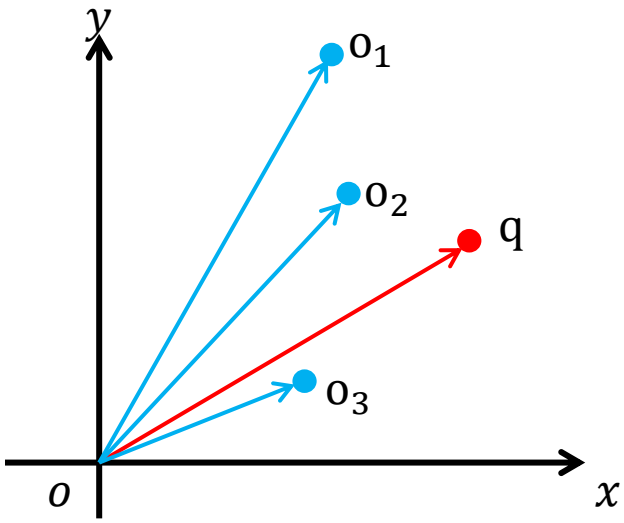
Meaning of Inner-Product

- Example
  - U_i stands for a user, I_j stands for an item, then R_{ij} stands for the relationship between U_i and I_j.
  - $$\begin{bmatrix} \cdots & \cdots & \cdots & \cdots \\ u_{i1} & u_{i2} & \cdots & u_{id} \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix} * \begin{bmatrix} i_{11} & i_{12} & \cdots & i_{1d} \\ i_{21} & i_{22} & \cdots & i_{2d} \\ \cdots & \cdots & \cdots & \cdots \\ i_{N1} & i_{N2} & \cdots & i_{Nd} \end{bmatrix}^T = \begin{bmatrix} \cdots & \cdots & \cdots & \cdots \\ r_{i1} & r_{i2} & \cdots & r_{iN} \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

What is MIPS?

- Definition
  - **MIPS(Maximum Inner-Product Search):** For a given dataset of n points $D \subset \mathbb{R}^d$ and a query $q \in \mathbb{R}^d$, *MIPS* problem is to efficiently find a point $o^* \in D$ such that

$$o^* = argmax_{o \in D} \langle o, q \rangle$$



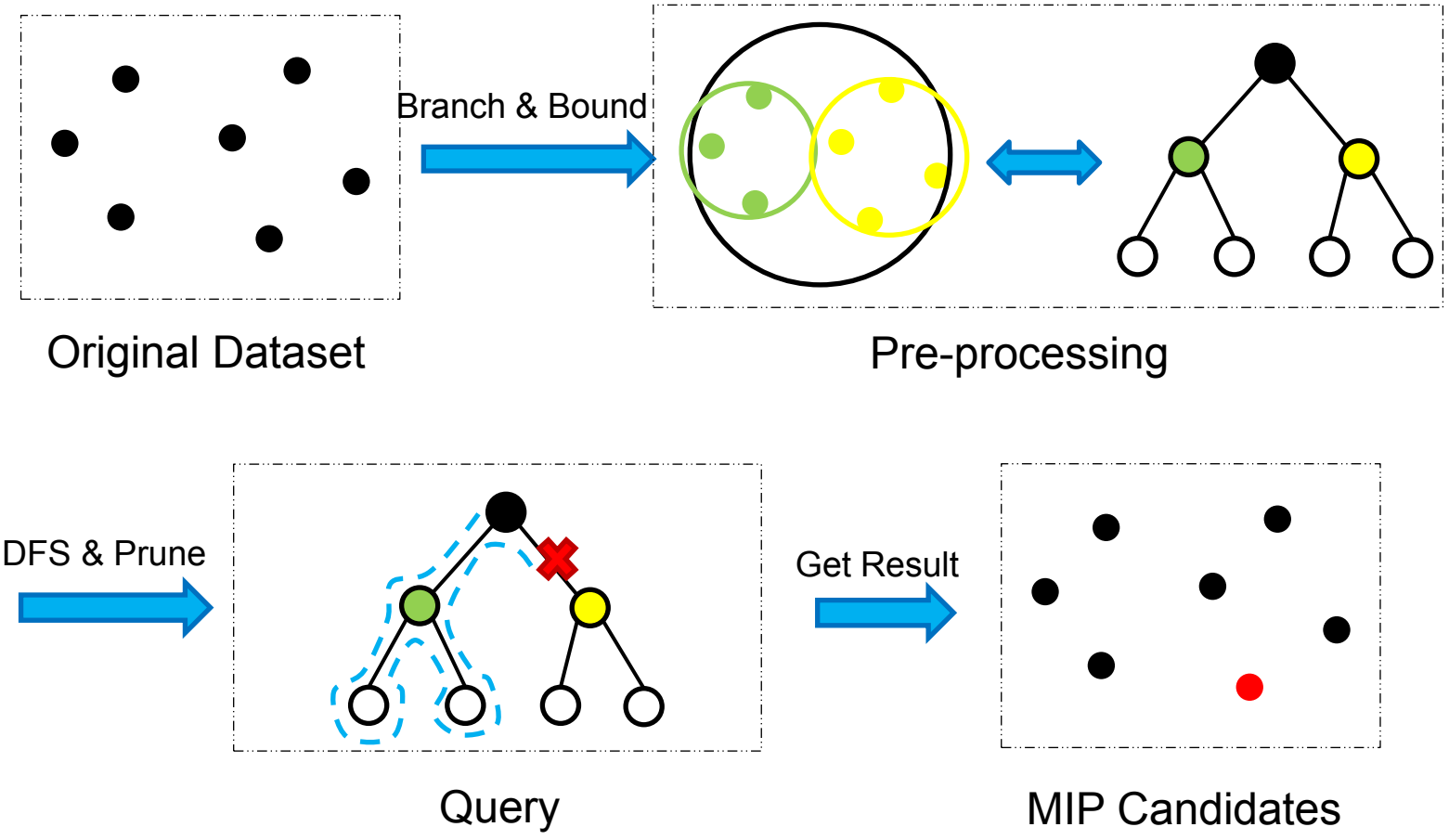
$$D = \{o_1, o_2, o_3\}$$
$$o^* = o_1$$

How does Ball-Tree Work?

- Framework

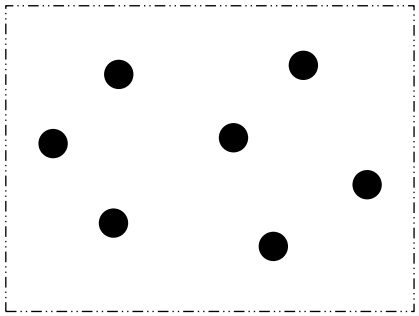
How does Ball-Tree Work?

- Framework



How to Branch?

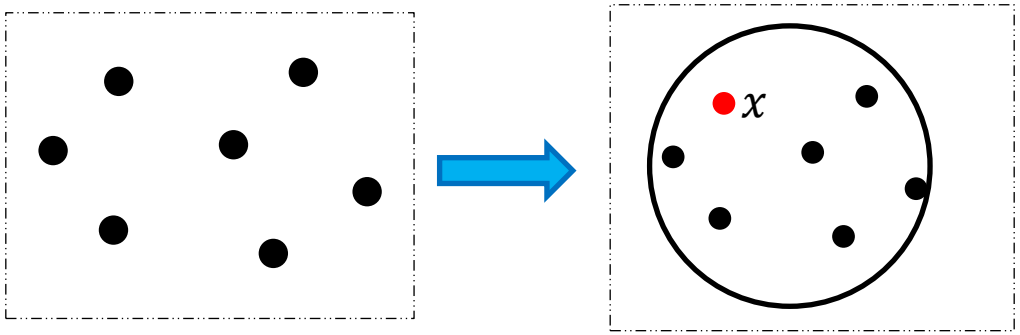
- Pre-processing
  - MakeBallTreeSplit



Original Dataset

How to Branch?

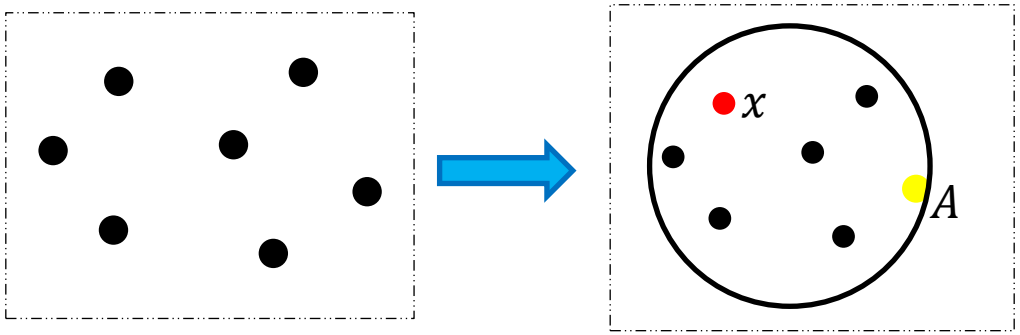
- Pre-processing
  - MakeBallTreeSplit
    - Firstly, pick a random point $x \in D$.



Original Dataset MakeBallTreeSplit

How to Branch?

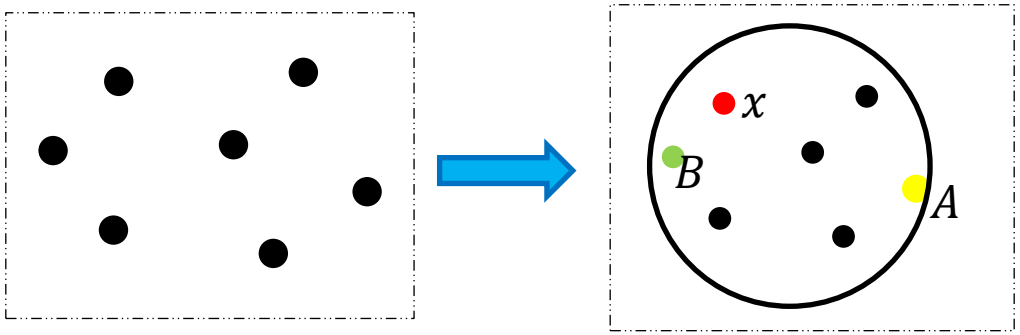
- Pre-processing
  - MakeBallTreeSplit
    - Firstly, pick a random point $x \in D$.
    - Then select the furthest point of x as A.



Original Dataset MakeBallTreeSplit

How to Branch?

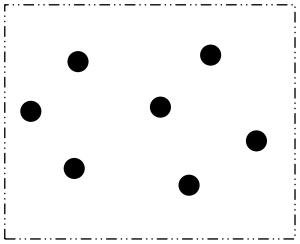
- Pre-processing
  - MakeBallTreeSplit
    - Firstly, pick a random point $x \in D$.
    - Then select the furthest point of x as A.
    - Finally, select the furthest point of A as B.



Original Dataset MakeBallTreeSplit

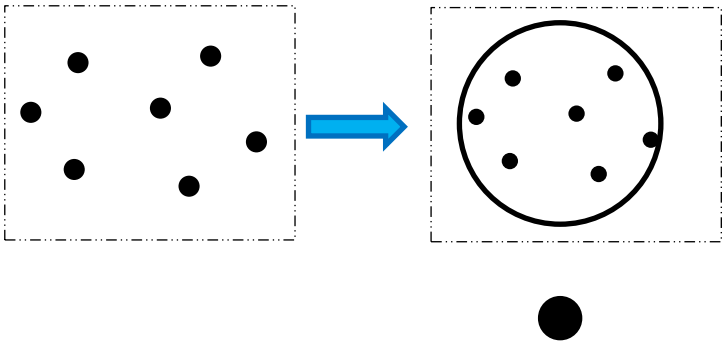
How to Branch?

- Pre-processing
  - MakeBallTree



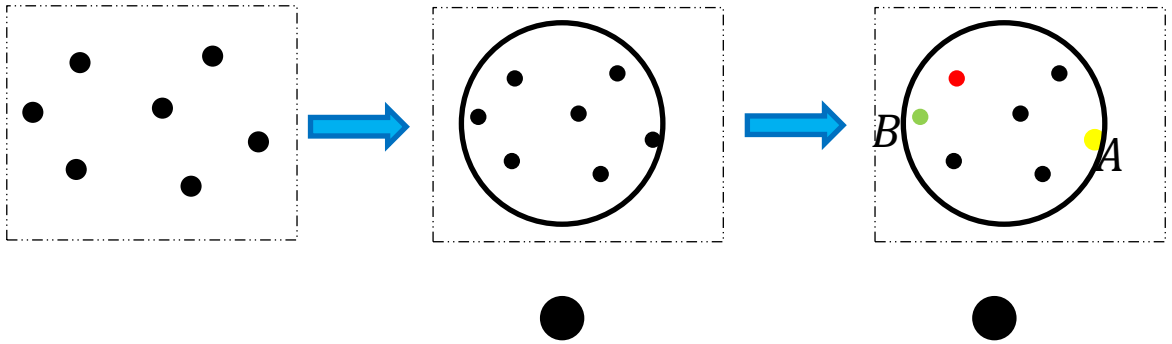
How to Branch?

- Pre-processing
  - MakeBallTree
    - Cover the dataset with a ball $B(C, R)$.



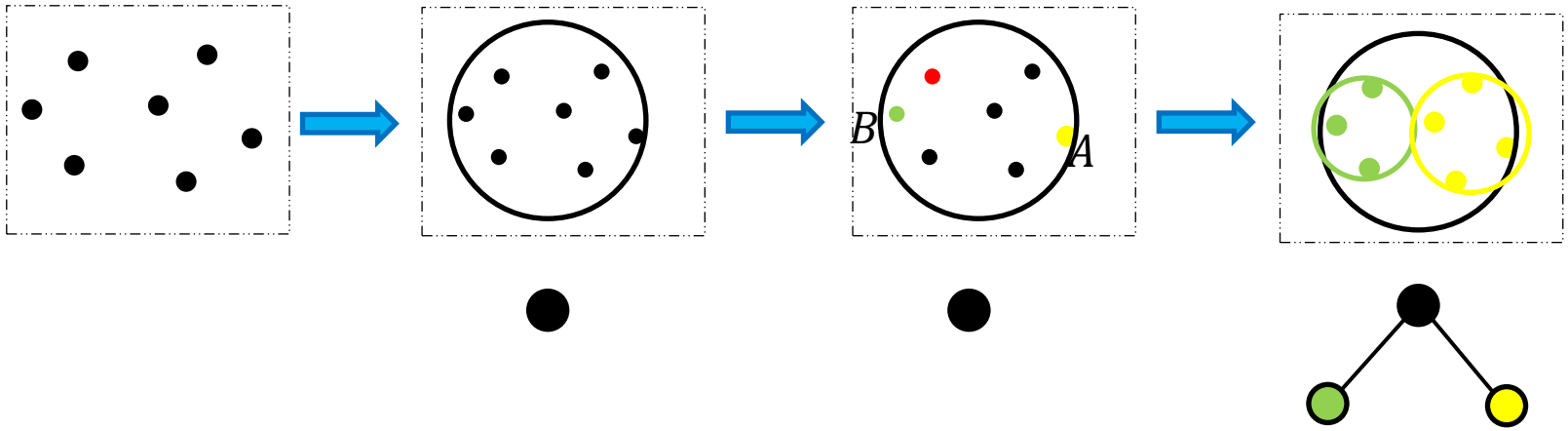
How to Branch?

- Pre-processing
  - MakeBallTree
    - Cover the dataset with a ball $B(C, R)$.
    - If $|D| \leq N_0$, stop splitting.
    - Else find (A, B) using MakeBallTreeSplit.



How to Branch?

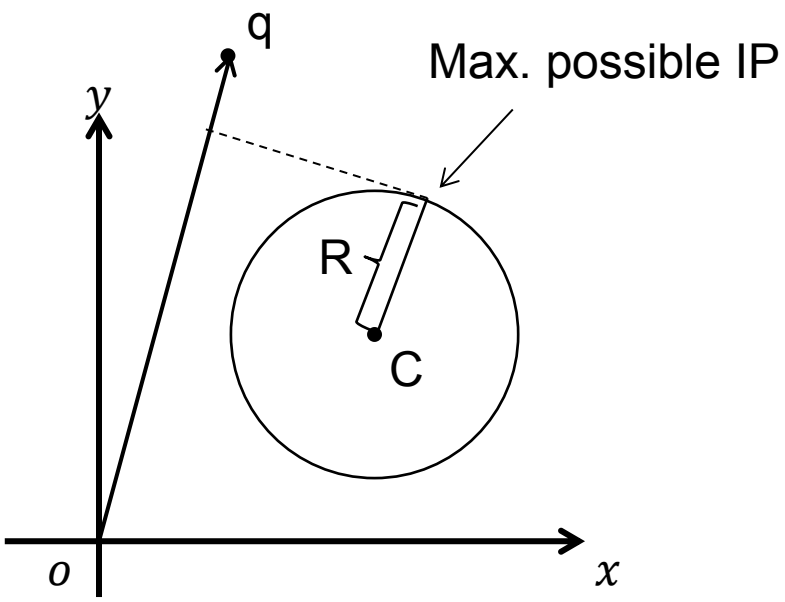
- Pre-processing
  - MakeBallTree
    - Cover the dataset with a ball $B(C, R)$.
    - If $|D| \leq N_0$, stop splitting.
    - Else find (A, B) using MakeBallTreeSplit.
    - Partition the dataset into two parts according to their distance to A and B.



How to Bound?

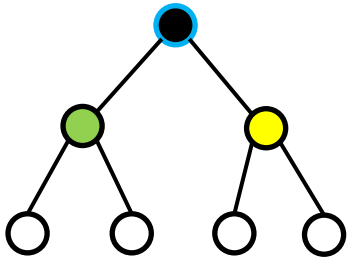
- Maximum Inner-Product Bound
  - For a tree node indexed with a ball $B(C, R)$ centered at C with radius R, the maximum inner-product possible between the query q and any point in the tree node is:

$$\max_{p \in B(C,R)} \langle q, p \rangle \leq \langle q, C \rangle + R \parallel q \parallel$$

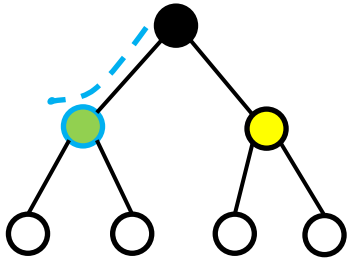


How to DFS?

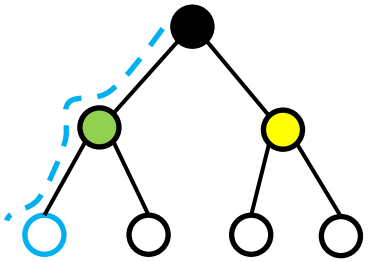
- Depth First Search



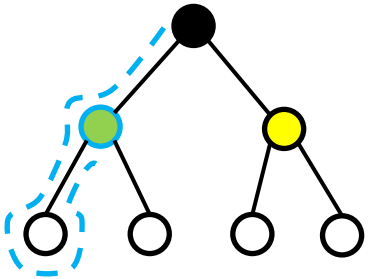
1.Select child @root



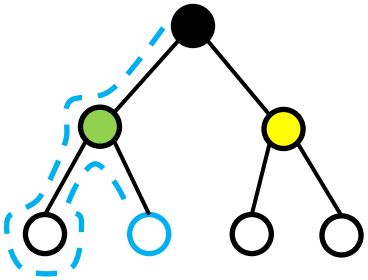
2.Recurse to best child till 1st leaf



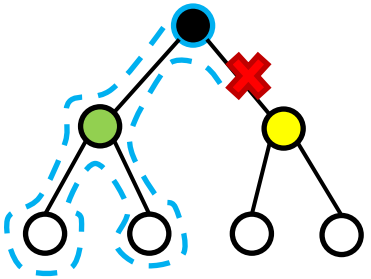
3.Obtain best candidate



4.Try to prune other children



5.Explore other children if pruning not possible



6.Save computation if pruning possible

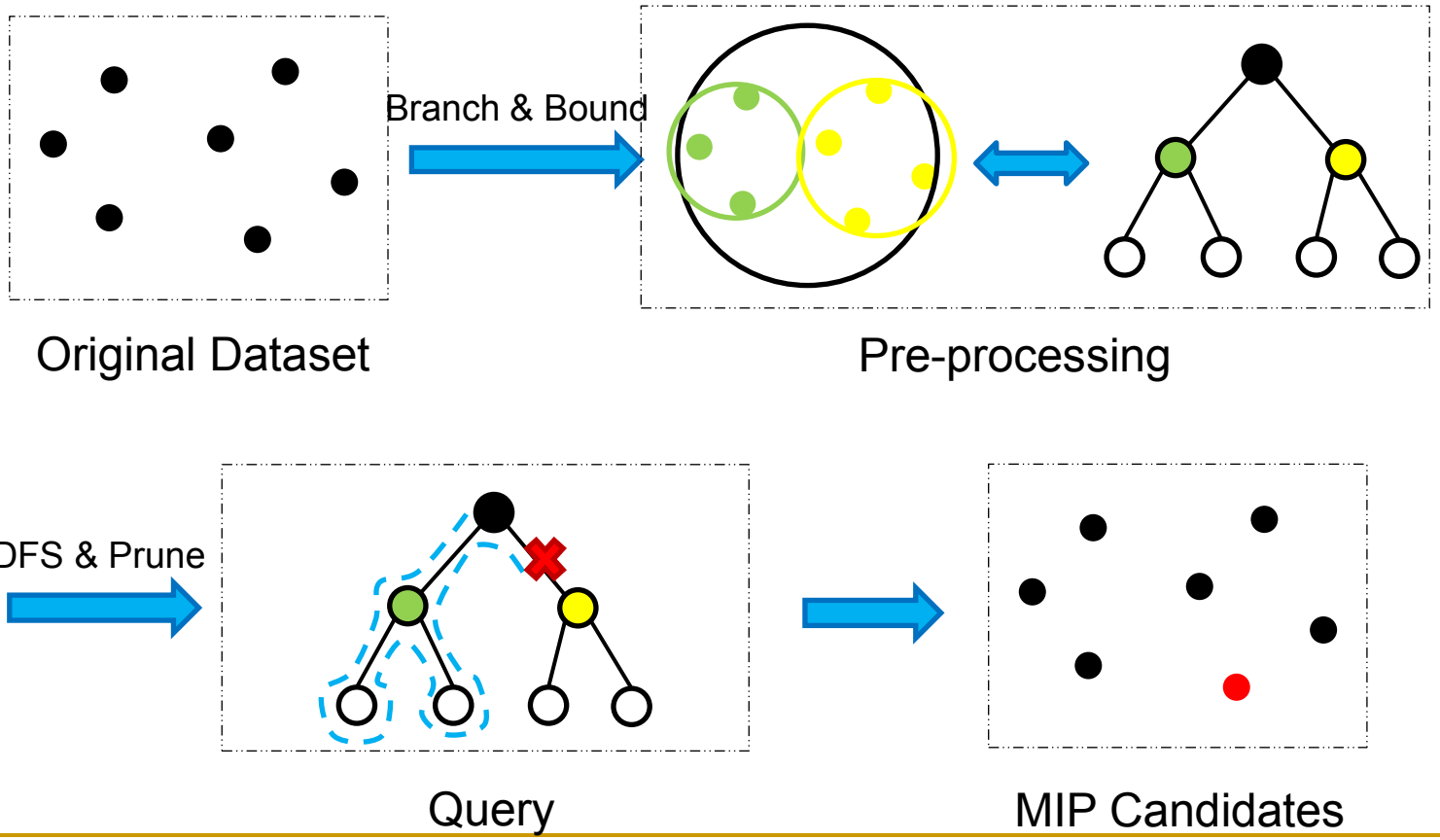
第三次作业

马桂洪，张楚涵

2017年5月11日

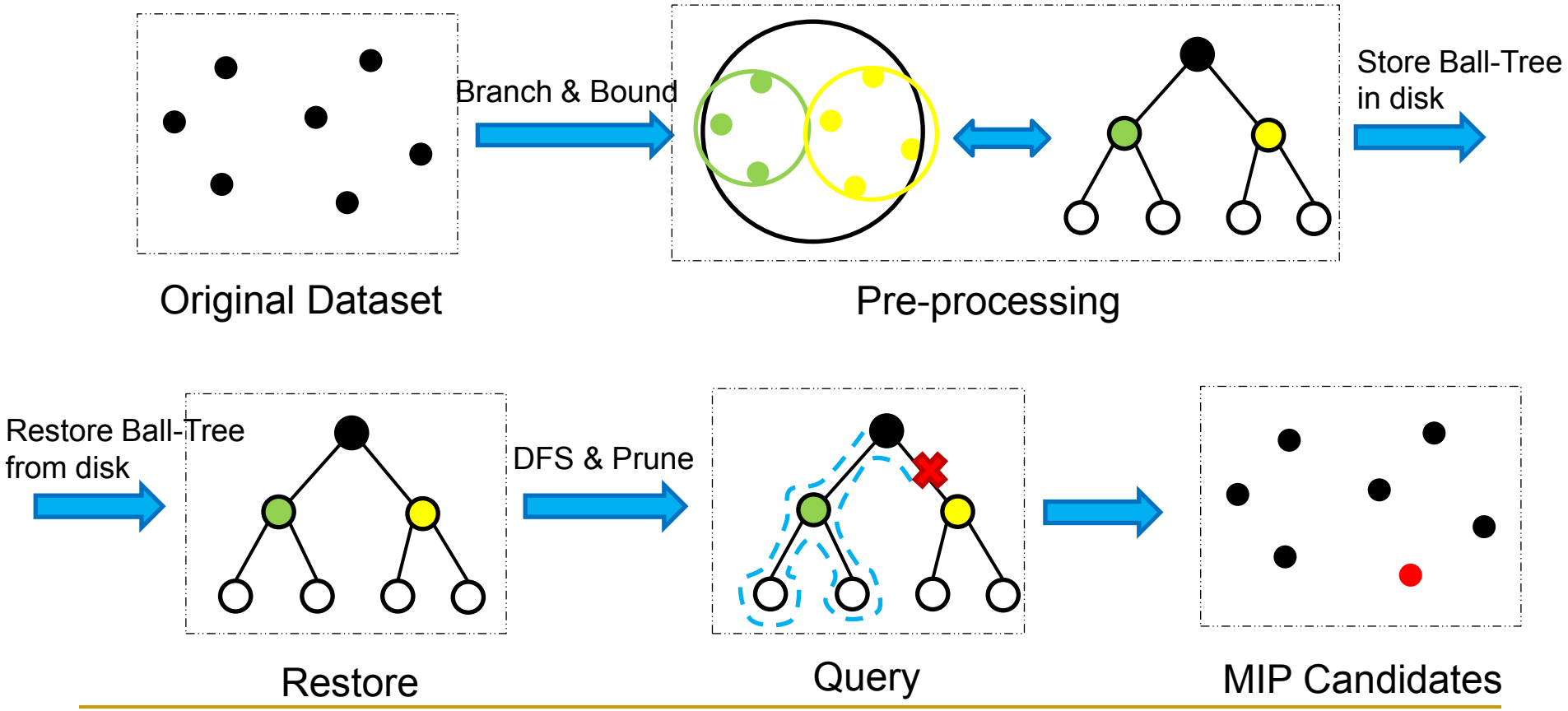
第三次作业

- Ball-Tree的算法框架



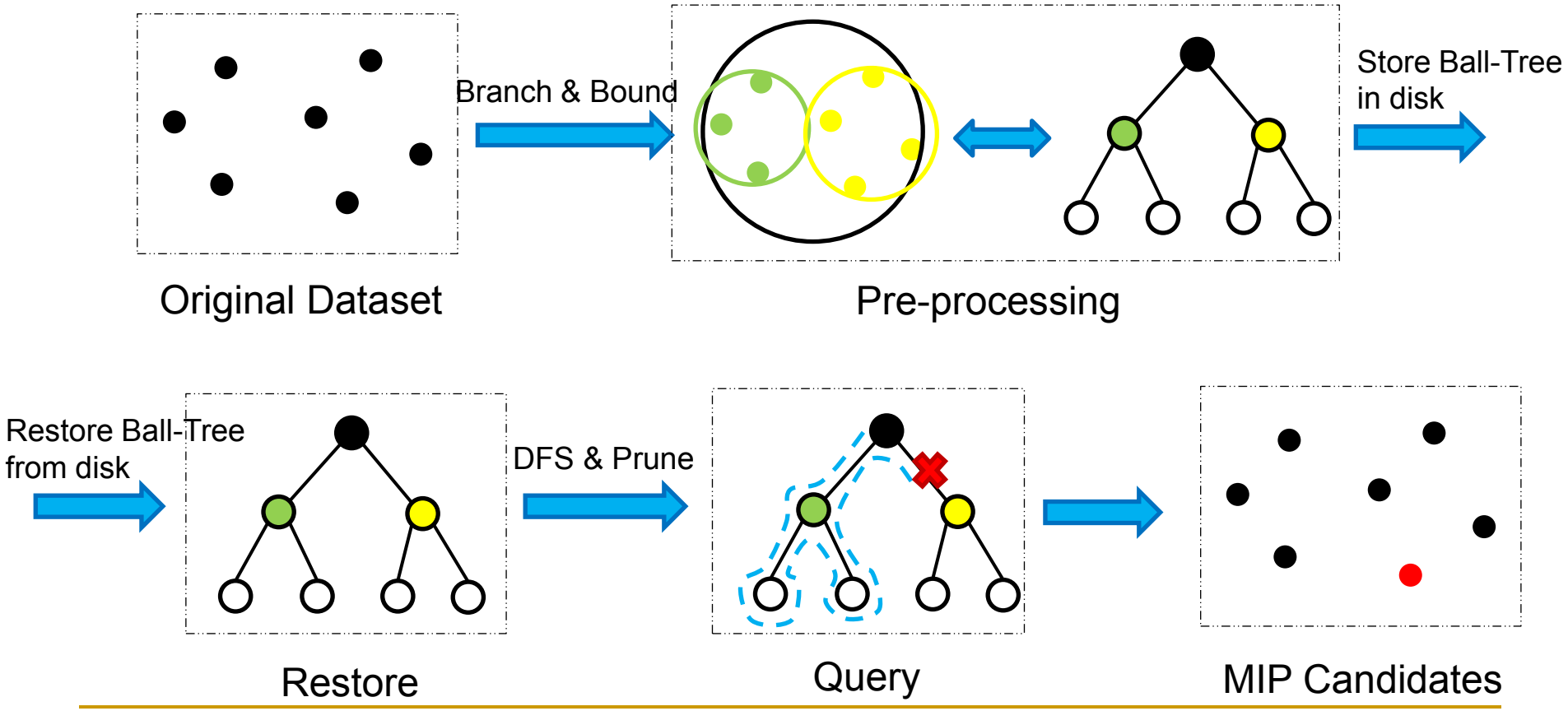
第三次作业

- 作业内容：实现Ball-Tree的C++外存版！



第三次作业

- 作业内容：实现Ball-Tree的C++外存版！



第三次作业

- 作业内容：实现Ball-Tree的C++外存版！
  - 任务1：实现ball-tree的建树过程（20分）
  - 任务2：实现将ball-tree写进外存的功能（30分）
  - 任务3：实现从外存中载入ball-tree的功能（20分）
  - 任务4：实现查询阶段找到最大内积对象并剪枝的功能（30分）

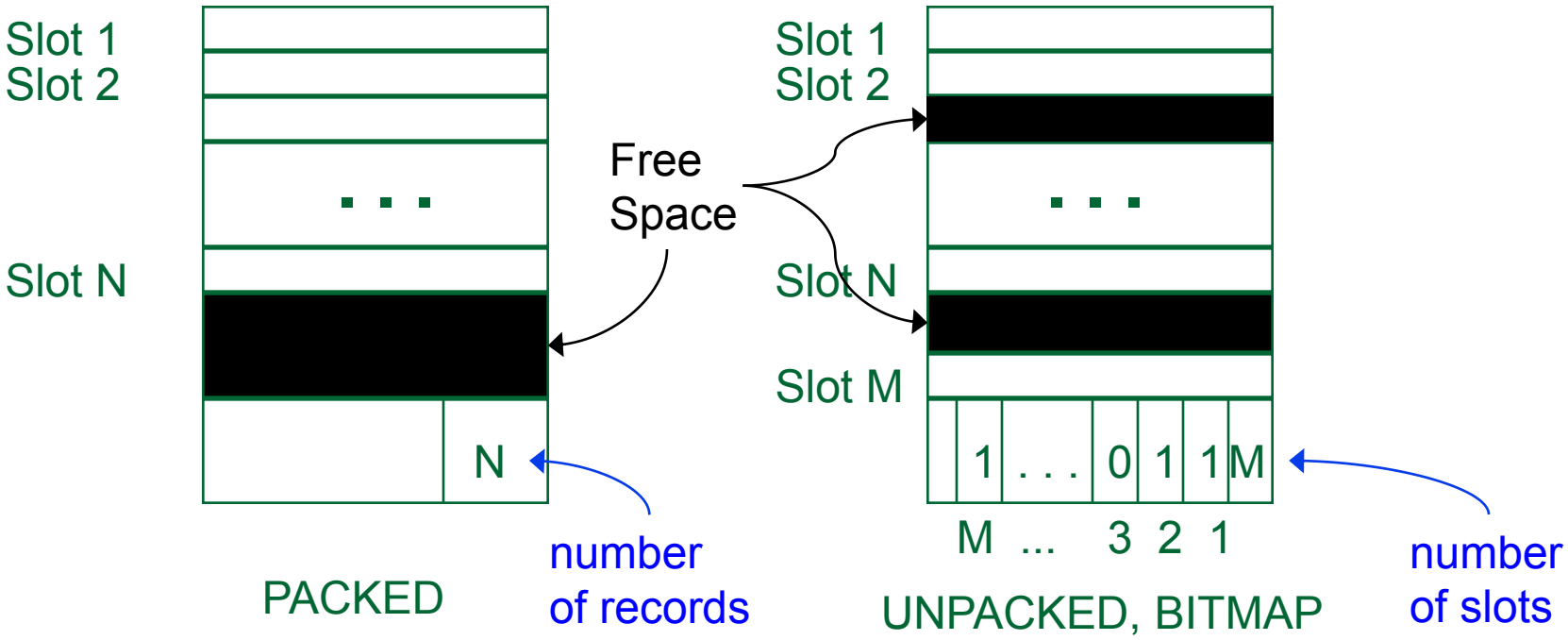
第三次作业

- 作业要求
  - 开发环境：linux
  - 任务1：设计数据结构，按照论文的伪代码实现即可。N_0设为20。
  - 任务4：深度优先搜索，按照论文的伪代码实现即可。

第三次作业

- 作业要求
  - 任务2：按定长记录存储（参考上课课件）
    - 要求按二进制的页格式存储。缓存页的大小设为$B = 64K$。
    - 每个槽存储一个树节点。
    - 给每个树节点指定ID，按树节点ID查找和存储。
    - 叶子结点和非叶子结点分开存储。（仅供参考）
    - 数据对象直接存放在叶子结点中。（仅供参考）
  - 任务3：
    - 不要将整棵树载进内存。
    - 用尽可能少的内存消耗，完成尽快的查询。

Page Formats: Fixed Length Records



- *Record id = <page id, slot #>. In first alternative, moving records for free space management changes rid; may not be acceptable.*

第三次作业

- 作业加分项（总分最多为100分）
  - 任务5：实现添加和删除数据对象的功能（15分）
  - 任务6：实现四叉ball-tree的功能（15分）
  - 注意：请在报告中写明实现过程

第三次作业

- 数据集
  - Mnist（仅供开发过程测试使用）

Datasets	#Objects	#Dimension	#Query	B	Data Size
Yahoo! Music	60,000	50	1000	64KB	9.4MB

  - Yahoo! Music（查询数据仅为样例，非最终查询数据）

Datasets	#Objects	#Dimension	#Query	B	Data Size
Yahoo! Music	624,961	300	1000	64KB	2.3GB

第三次作业

- Source
  - 不完整的代码框架：

  - 数据集和查询集样例：

第三次作业

- 提交内容
  - 完整的代码

  - 小组报告
    - 命名格式：TeamID_homework3_report.pdf

第三次作业

- 提交格式
  - 所有文件放在BallTree文件夹下
  - 打包后以rar或zip文件提交
  - 命名格式：TeamID_homework3.zip
- 截止日期
  - 2017年5月31日23：59