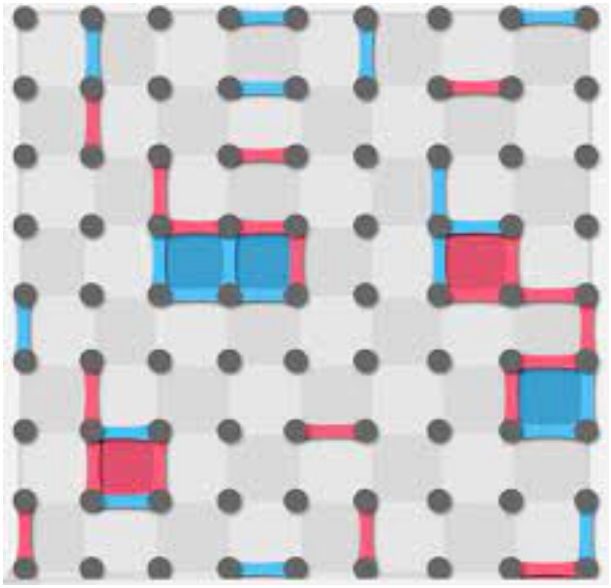
Data Systems

Assignment - 1 (Submission-2)

Dots and Boxes



Team Information

Name	Roll No	Email
Anurag Gupta	2020101019	anurag.g@students.iiit.ac.in
Madhusree Bera	2022202007	madhusree.bera@students.iiit.ac.in
Sanya Gandhi	2022201066	sanya.gandhi@students.iiit.ac.in

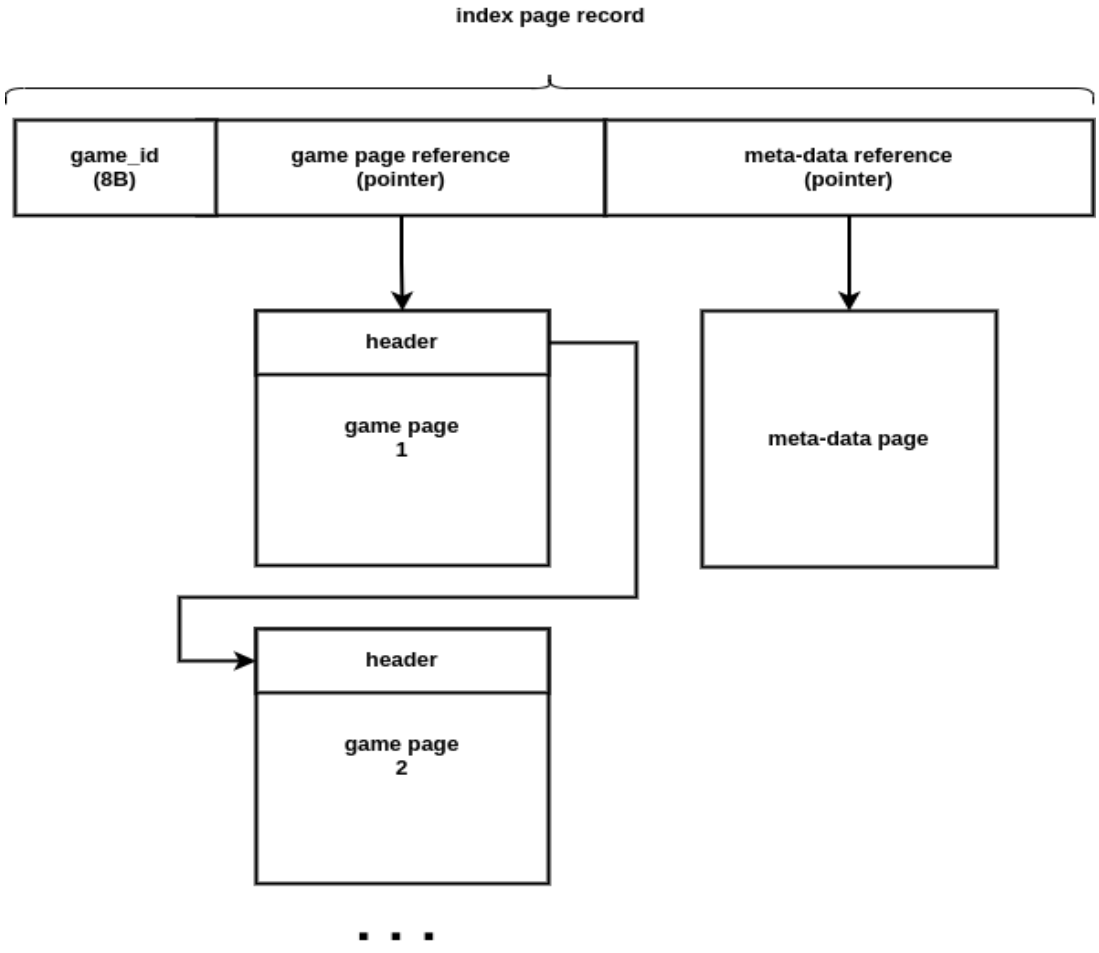
Assumptions

1. Game related assumptions
    1.1. Players are given a turn according to the sequence of Player IDs.
    1.2. The move made by the player is provided to the database in the form of tuples containing the information (index of the point from where the line started, the direction of the line drawn (U,D,R,L))
    1.3. There are no illegal moves.
2. Software related assumptions
    2.1. The software checks if a box has been created and provides another opportunity to the player that completed the box.
    2.2. On recreating the game, the software keeps track and identifies which player's turn it is to make the move based on the previous moves.
    2.3. The software provides the order of players.
    2.4. Software provides the number of boxes acquired by each player and the winner info at the end of the game.
3. Parameter assumptions:
    3.1. Minimum number of players:100, maximum:256
    3.2. Size of the game:1000 x 1000
    3.3. Page size: 4KB
4. The config file consists of
    4.1. Size of the game: m and n
    4.2. Number of players
    4.3. Number of games played

    For each config file, a new index page will be created.

Proposed Methodology

1. The order of the player is stored in the Meta-data page which is accessed through the Index page of the game.
2. Every move made in the game is logged into the database in a sequential manner in the form (starting_dot_index, direction_of_line, box_fill_flag).
3. Every point/dot in the game is assigned an index starting from 0 to 10^6 -1.
4. Every move needs to be logged sequentially in the order it was played in the game.

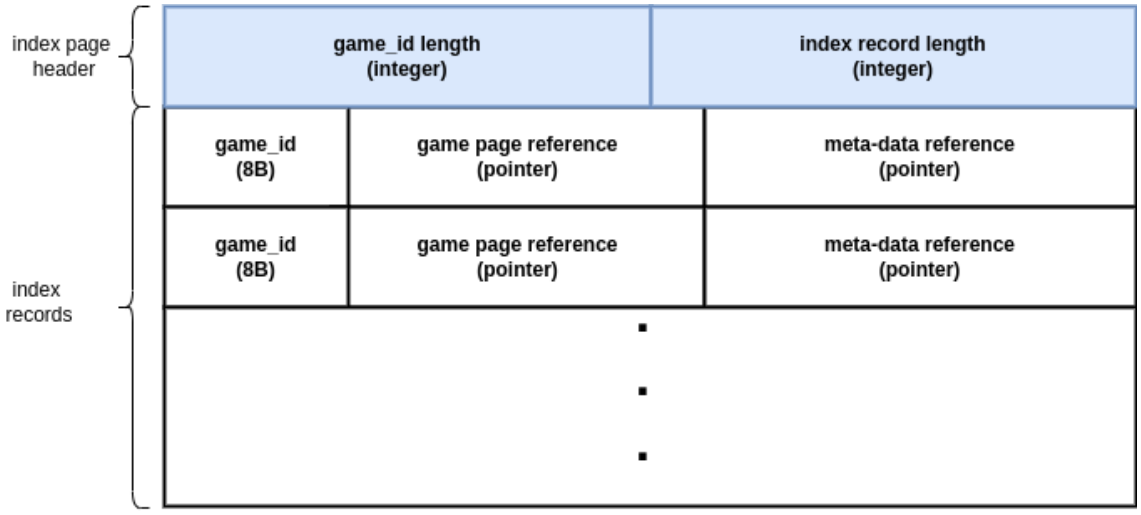


Page design overview

Page Design

1. The proposal include three categories of pages
2. All the three types of pages comprise page header and game information.
3. The 3 types of pages are: Index Page, Game Page and Meta-Data Page

3.1. Index Page



index page design

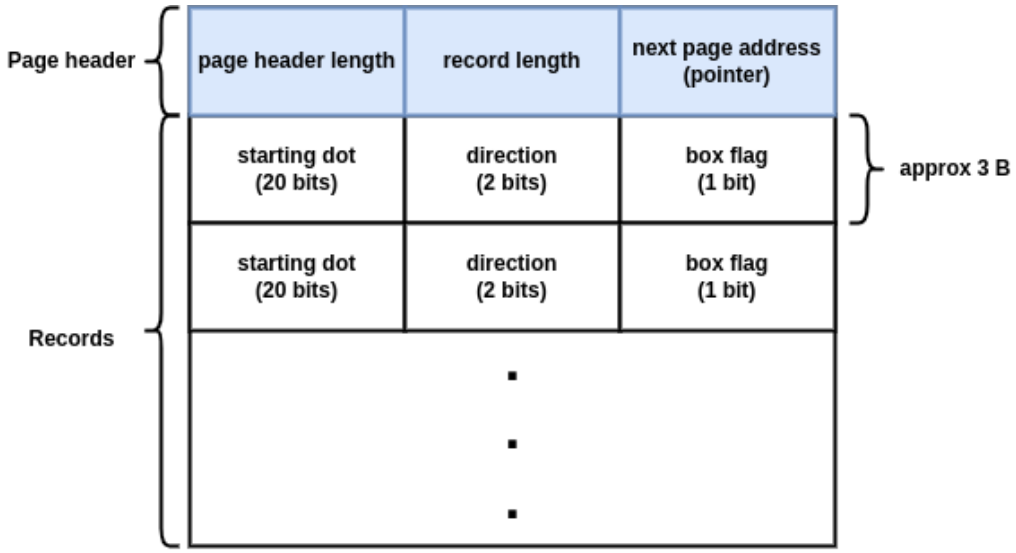
3.1.1. Index page as the name suggests is used for indexing the game and meta-data pages for a given game ID. It has three fields: Game ID, memory referencing address of the first- Game page of each game and of the Meta-data of each game.

3.1.2. Multi-level indexing of index pages for each config file can be done for optimizing search of game ID in index pages.

3.1.3. The game_id can vary between 0 and 10^{18} -1. To accommodate this, we need $\log_2(10^{18}-1)$ bits that is approximately equal to 60 bits which needs allocation of 8 Bytes to this information.

3.1.4. The index page header contains 2 integers: game_id length and index record length. This will help in fetching the index records from the index pages.

3.2. Game Page



game page design

3.2.1. The Game page has the index of the first dot from where the line is drawn, the direction in which the line is drawn and box_fill flag which tells if a box was created in the respective move.

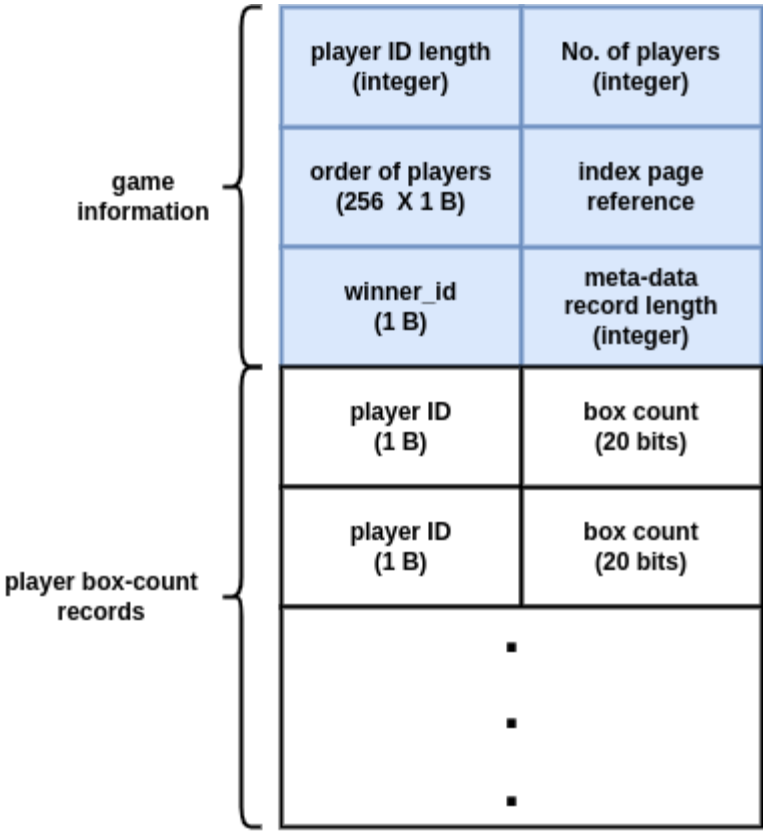
3.2.2. The Game pages are the pages that include move's information in sequential manner. The address of the first game page as mentioned above is stored in the index page and every game page has the memory address of the next game page for that particular game. The last game page will have a null pointer stored in the next game page address field.

3.2.3. The starting dot index will be 20 bits long (acquired as $\log_2 10^6$ bits), 2 bits of direction (right,left,top,bottom), and 1 bit for box_fill_flag.

3.2.4. Every entry in the game page design hence approximately needs 3 Bytes.

3.2.5. The game page header contains : header length, record length and next page address. These fields will help in reading the records from game page and go to the next game page.

3.3. Meta-data page



Meta-data design page

3.3.1. Meta-data page has three sections: Header, Player to box-count mapping and winner id information. Every game has one corresponding Meta-data page.

3.3.2. Its header has player ID length, No. of players which will tell the number of records in the meta-data page, order of players, memory reference to the index page, length of each record which will help in fetching the records.

3.3.3. The mapping section stores player IDs (1Byte for each ID) and the corresponding number of boxes (represented by 20 bits) acquired throughout the game.

3.3.4. The winner id (1Byte) information is also present in this page which can be accessed in the header.

4. Importance of box_fill_flag field in space optimization:

4.1. For the mapping of players with the moves, player_ID along with the moves is required to be stored.

4.2. Instead of storing the player_ID, box_fill_flag field provides multiple information regarding the player as well as filling of boxes.

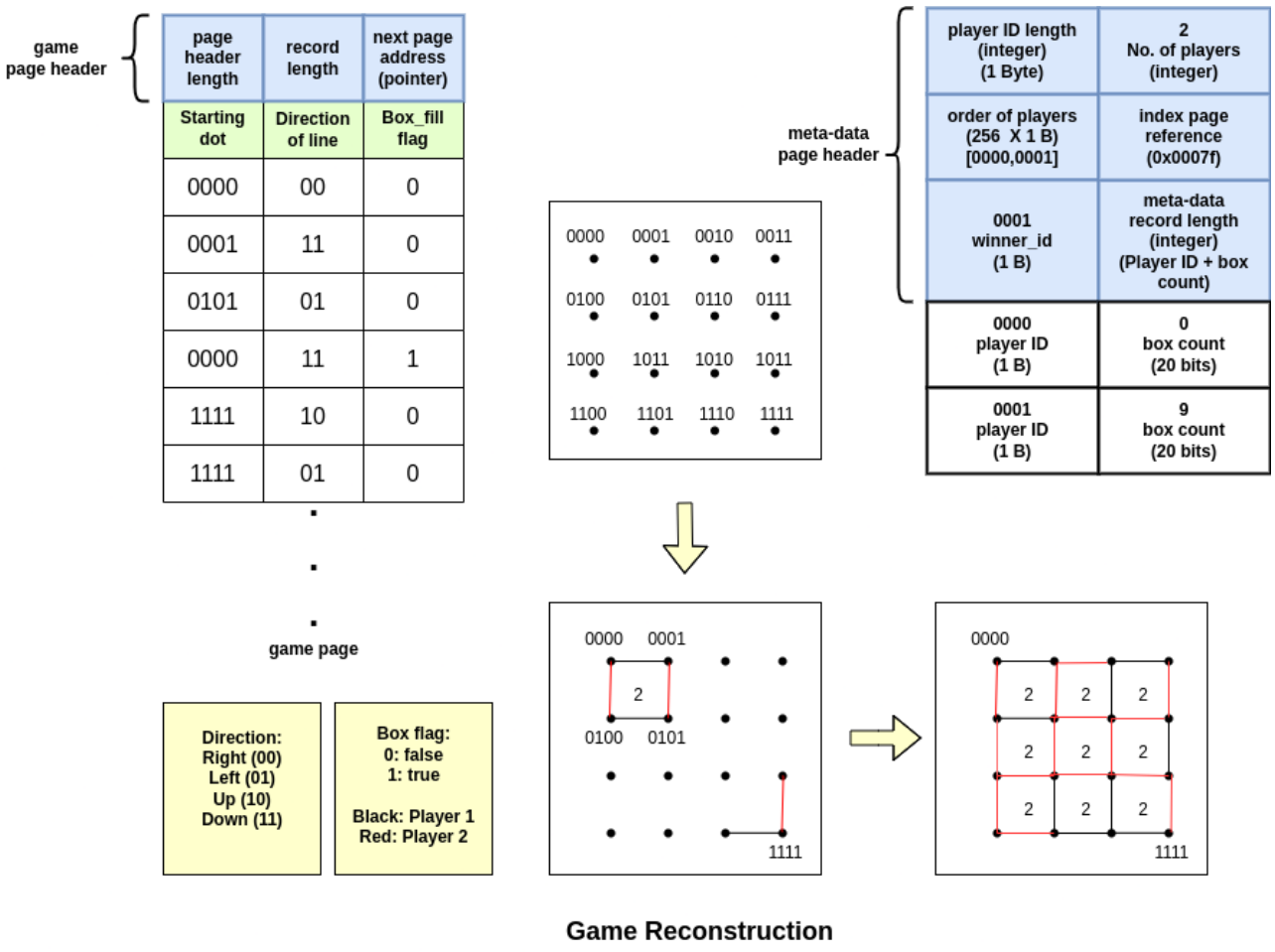
4.3. Once the order of players is known by the software, it can map the moves with players in such a way that if the box_fill_flag field is true, the next move will be assigned to the last player.

4.4. This way, we are saving almost 2MB of space per game as player id per move would have consumed 1B of space and there are $2*10^6$ Bytes of moves. Eventually for the move information, 8MB could have been used but now 6MB is used.

4.5. Also, saving box_filled status, a simple query can now calculate number of boxes filled by each player in O(n) time.

Steps for game reconstruction:

1. Access the Index page of the game.
2. Reach Meta-Data page ▶️ Access player order.
3. Access moves' information from Game pages.
4. Get Player to box-count mapping from Meta-data page.
5. Fetch Winner ID from Meta-data page.



Game Reconstruction

Features

- Since the data is being stored in a sequential manner, we do not need to store the player ID of the player who made the move as that can be computed in runtime by the game program.
- Each game consumes approximately 6MB of storage space for the given config file.
- The box_fill_flag enables efficient determination of the next player's turn and the box-count of each player .
- The moves information pages link to the next page which makes the design scalable. When the size of the game increases, this design can easily accommodate new pages as more moves are added.
- Since we are storing the address of the first game page, this is making our design space efficient.
- For every config file, a new index page will be created and corresponding game and meta-data pages will be generated. The information provided in the header of these pages will be sufficient for reconstruction of any game for a given game_id.
- The design is modular and hence serves the purpose of scalability.

Updates in Submission 2

- Multi-level indexing of index pages for optimizing game_id search.
- Adding a new index page for a new config file as it will have new game_id length and length of other parameters will change as well.
- Meta-data page is also shown for the given 4x4 game reconstruction example.
- No other changes have been made in the page design.