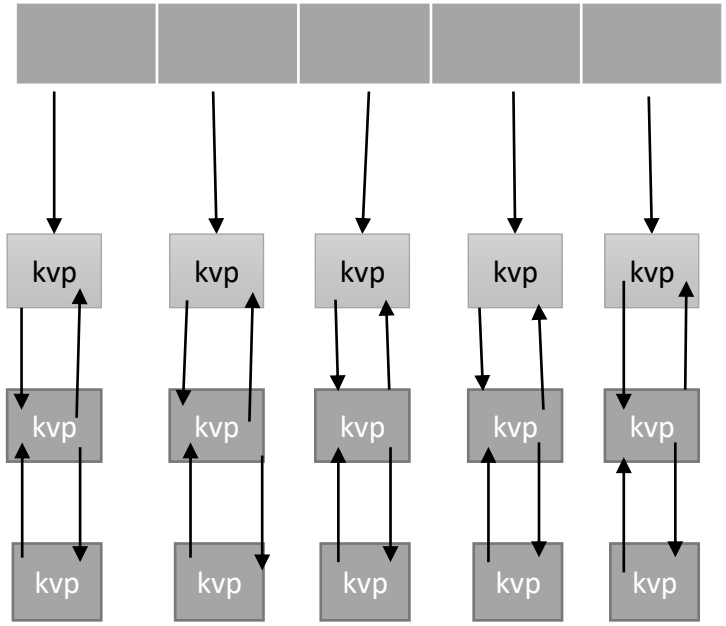
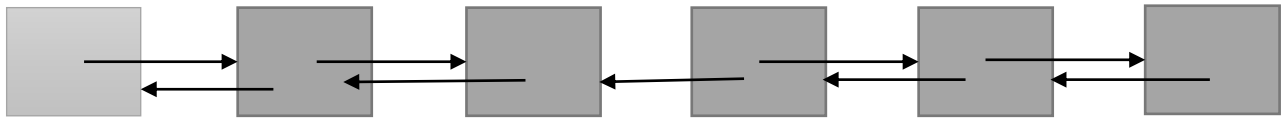


Jerries

Linked list of all the jerries

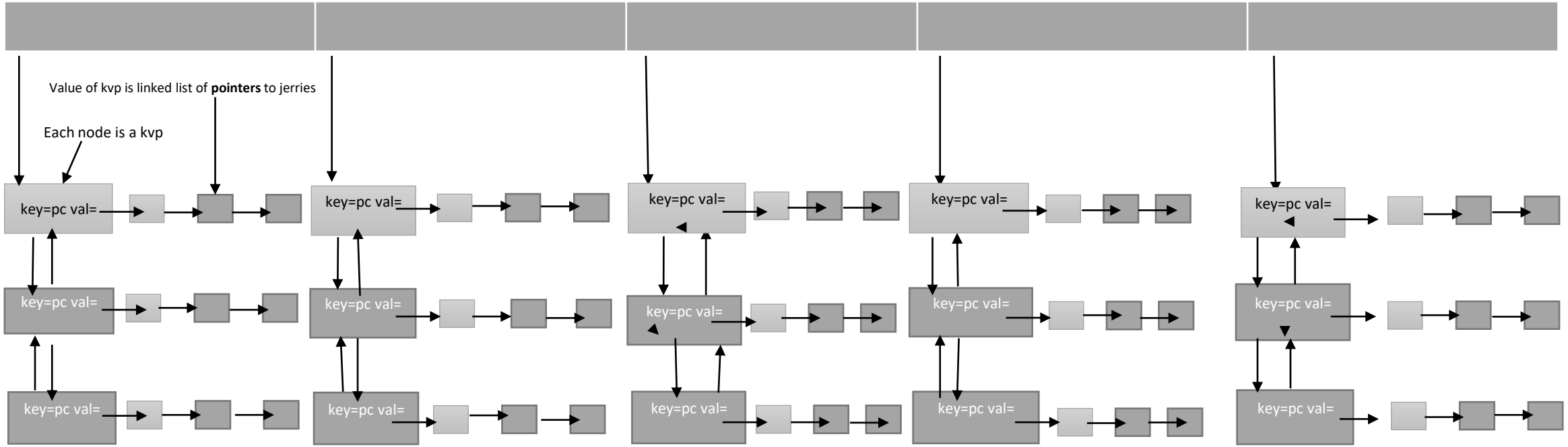


Hashtable JerryHT

Array of linked lists that every node is a kvp that the key is jerry id and value is a **pointer** to the jerry object itself

Multivaluehashtable PcMvht

Array of linked lists that each node is a kvp that its Key is Pcname(name of physical characteristic) and it Value is a linked list of **pointers** to jerries that have that pc



JerryBoree:

בחרנו לממש את הjerryboree שלנו בעזרת שלושה מבנים כפי שאפשר לראות בדף הראשון. בחרנו במבנה jerries כמבנה היחיד עם פעולת מחיקה ופעולת העתקה של ג'רים. עשינו זאת מכיוון שבשאר המבנים אנו מחזיקים פויינטרים לאותם ג'רים במבנה jerries כדאי למזער בשימוש מקום ולמקסמם יעילות חיפוש וגישה, ולא נרצה שבמבנים האלה תיהיה את האשפרות למחוק ג'רים אלה רק לשחרר את אותו פויינטר. בכך אנו מבטיחים שכמות הג'רים בכל המבנים שלנו יהיה ככמות הגרים ולא יותר.

Jerries :

מבנה זה הוא המבנה שנשמור בו את כל הג'רים שלנו מהקובץ קונפיגורציה ואת כל הג'רים שיתווספו. הADT שבחרנו הוא הלינקד ליסט שמימשנו שבו כל node מחזיק כערך ג'רי . בחרנו במבנה זה שיהיה המבנה היחיד שישמור את הג'רים ויהיה נוח לגישה ולמעבר על כל הג'רים לפי סדר הכנסה.

JerryHT :

את המבנה הזה בחרנו לממש בעזרת הHashtable ADT אשר מימשנו. טבלת גיבוב זו תחזיק מערך של לינקד ליסטים (בשביל ה chaining) כאשר בכל לינקד ליסט (לפי פונקציית המיפוי שלנו) כל קודקוד הוא KEYVALUEPAIR ADT שגם מימשנו. לכל KEYVALUEPAIR במבנה זה המפתח הוא הid של ג’רי והערך הוא מצביע לאותו מקום כמו הג’רי במבנה הראשי שלנו Jerries המתאים למפתח ה ID היחודי של אותו ג'רי. בחרנו במבנה זה מכיוון שרצינו גישה בזמן ריצה ממוצע של O(1) לכל ג’רי.
באפשרויות ת1,4,6 בתפריט נשאלת השאלה מה הid של הג'רי ונרצה לבדוק אם קיים כזה במערכת ולגשת לג’רי זה בזמן ריצה הנדרש.
גודל המבנה יהיה המספר הראשוני הראשון אחרי כמות הג'רים (שנודע מגודל jerries) שקיבלנו בקובץ הקונפיגורציה. עקב כך נניח התפלגות אחידה וכתוצאה מכך חיפוש ג'רי יקח בממוצע O(1) כנדרש בסעיפים אלה.

PcMvht :

את המבנה הזה בחרנו לממש בעזרת הMultiValueHashtable ADT אשר מימשנו. טבלת גיבוב זו תחזיק מערך של לינקד ליסטים (בשביל ה chaining) כאשר בכל לינקד ליסט (לפי פונקציית המיפוי שלנו) כל קודקוד הוא KEYVALUEPAIR ADT שגם מימשנו. לכל KEYVALUEPAIR במבנה זה המפתח הוא שם של physical characteristic והערך הוא מצביע מצביע ללינקד ליסט. כל קודקוד בלינקד ליסט יחזיק מצביע לג'רי אשר יש לו את אותה תכונה.
בחרנו לבנות את המבנה הזה כך כדי שכאשר התפריט מבקש physical charactersitic נוכל להחזיר ב(O(1 ממוצע האם יש ג'רים במערכת עם תכונה זו (מבנה זה יעמוד בזמן בהנחה שיש התפלגות אחידה). דוגמא לצורך זה הוא כאשר בוחרים בסעיף 7 בתפריט. בנוסף נרצה לעמוד בזמן ריצה
O(number of Jerries with physical characteristic + total number of their physical characteristics) שדורשים בסעיפים 2 ו5.
במבנה זה כאשר מקבלים כקלט physical characteristic ניתן לעבור על כל הלינקד ליסט ולדעת לאיזה ג'רים יש את התכונה הזו. פעולה זו תקח
O(1) + O(number of jerries With physical characteristic). כאשר אנו מתבקשים להחזיר את הג'רי עם הערך של אותו physical characteritic שהכי קרוב לזה שהכנסנו כמו בסעיף 5 נצטרך לעבור על הלינקד ליסט ולעבור על כל תכונות של כל ג'רי ולבדוק איזה מהם הערך שלו הכי קרוב לערך שהוכנס. פעולה זו תיקח בדיוק:
O(number of Jerries with physical characteristic + total number of their physical characteristic) מכיוון שאנו עוברים על הלינקד ליסט בזמן:
O(number of Jerries with physical characteristic) בנוסף לכל ג'רי ברשימה נעבור על כל התכונות שלו בכדי למצוא את הערך של אותה תכונה שזה יקח:
O(total number of the jerries physical characteristics) שביחד יקיים את הזמן ריצה הנדרש. כנ"ל בסעיף 2 נצטרך לחפש האם יש לגרי את התכונה שהוכנסה ואם לא להוסיף אותה, כלומר נלך ל **PcMvht** , נחפש את התכונה והאם הג'רי שהוכנס קיים בלינקד ליסט זה שיקח: O(number of Jerries with physical characteristic)
ולבסוף תצטרך להדפיס את כל הג'רים בעלי תכונה זו שיקח O(number of Jerries with physical characteristic + total number of their physical characteristics).
כלומר, סך הכל קיבלנו זמן ריצה של O(number of Jerries with physical characteristic + total number of their physical characteristics) לחיפוס והדפסה כנדרש.