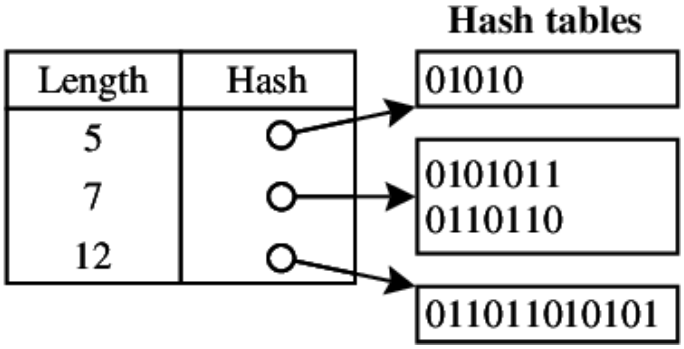
Q1- Packet classifier

c. נבנה trie כמו בסעיפים הקודמים אך כעת נרחיב כל node כך שכל node יכיל 4 ביטים בתוכו ולא ביט בודד.
הרחבת כל node מאפשרת לדחוס את המסלולים דבר המוביל לחיפוש קצר יותר מאחר וגובה העץ המקסימלי שיכול להיות קטן, בעצם אנו משווים כל פעם מספר ביטים ולא ביט בודד דבר המייעל את החיפוש ומקצר את הזמן והגישות לזיכרון.
עד כה במקרה הגרוע בגובה המקסימלי של העץ היה 32 כמספר הביטים הכולל בכתובת IP, ואילו כעת גובה העץ המקסימלי במקרה הגרוע הינו 8 מאחר וכל node בעצם מחזיק בתוכו 4 מידע על 4 ביטים.
כלומר במקרה הגרוע ביותר נעשה חיפוש לאורך כל העץ, סה"כ 8 lookups.

כנ"ל לגבי כל מספר אחר של Lookups שנרצה. במידה ונרצה לדוגמה 4 lookups אז נרחיב כל node כך שישמור 8 ביטים בתוכו ובעצם נצמצם את גובה העץ לעץ בגובה 4 לכל היותר.

d. דרך אלטרנטיבית ליצירת prefix table: במקום השימוש בעץ נשתמש בHash table
לכל אורך אפשרי של prefix ניצור hash table היכיל בתוכו את כל הprefix-ים הקיימים בגודל זה.
בעת ההכנסה נכניס למקום לפי אורך הprefix של כתובת הIP. כנ"ל גם לגבי מחיקה.
בעת החיפוש – נתחיל לחפש מאורך האורך ביותר, אם מצאנו התאמה אז נחזיר אותה, זאת תהיה ההתאמה הארוכה ביותר. אם לא, נסתכל על האורך הראשון הקטן יותר (כלומר בדוגמה מטה נחפש כעת ב7) במידה ומצאנו התאמה נחזיר אותו וכן הלאה.

לדוגמה-



היתרון – בעת הכנסה/ מחיקה של Prefix יותר מהיר ויעיל.

חסרון- יכול לקחת הרבה מקום, בעת החיפוש יכול לקחת כאורך הprefix.