Lab 01 CTC-12

Rafael Camargo

Professores

Luiz Gustavo Bizarro Mirisola

Instituto Tecnológico de Aeronáutica – ITA

1. Questão 1

1.1. (pergunta mais simples e mais geral) porque necessitamos escolher uma boa função de hashing, e quais as consequências de escolher uma função ruim?

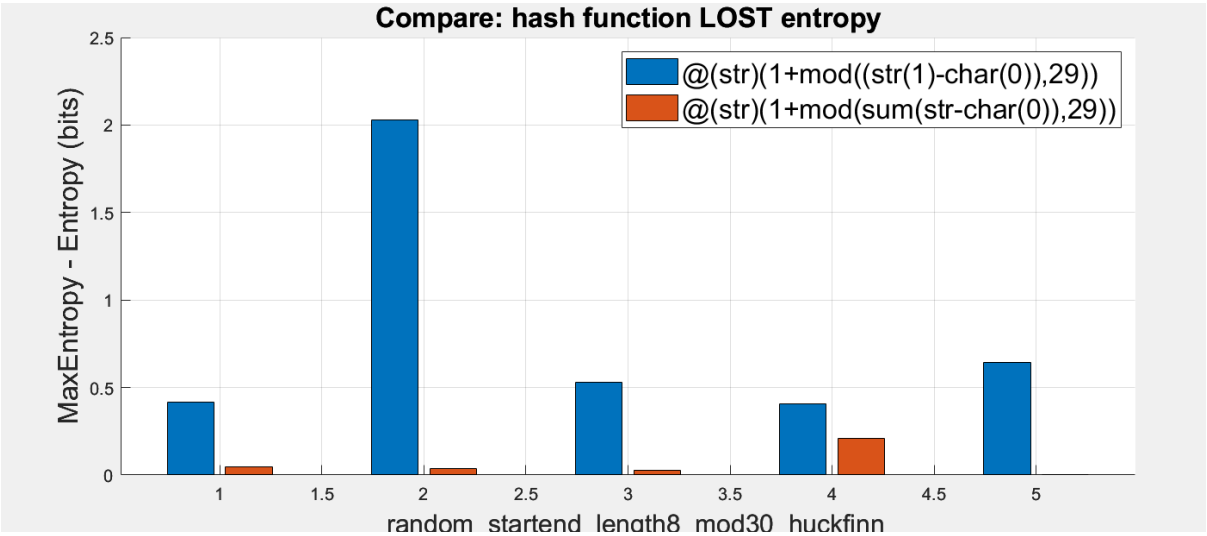
É necessária uma boa função hash para que um conjunto aleatório suficientemente grande de dados seja armazenado de maneira homogênea ao longo da hash table. Caso uma função ruim seja escolhida, a distribuição pode não ser homogênea.

1.2. Porque há diferença significativa entre considerar apenas o 1o caractere ou a soma detodos?

Pois a soma de todos os caracteres é um conjunto maior de dados e utilizar apenas o primeiro caractere pode criar viés do local que a string vai ser armazenada. Como no conjunto de dados “startend”.

1.3. Porque um dataset apresentou resultados muito piores do que os outros, quando consideramos apenas o 1o caractere?

Pois esse conjunto de dados possui um viés alto. Ao analisar o conjunto “startend”, é possível notar, por exemplo, que existe uma grande quantidade de palavras que começam com C.



2. Questão 2

2.1. Com uma tabela de hash maior, o hash deveria ser mais fácil. Afinal temos mais posições na tabela para espalhar as strings. Usar Hash Table com tamanho 30 não deveria ser sempre melhor do que com tamanho 29? Porque não é este o resultado?

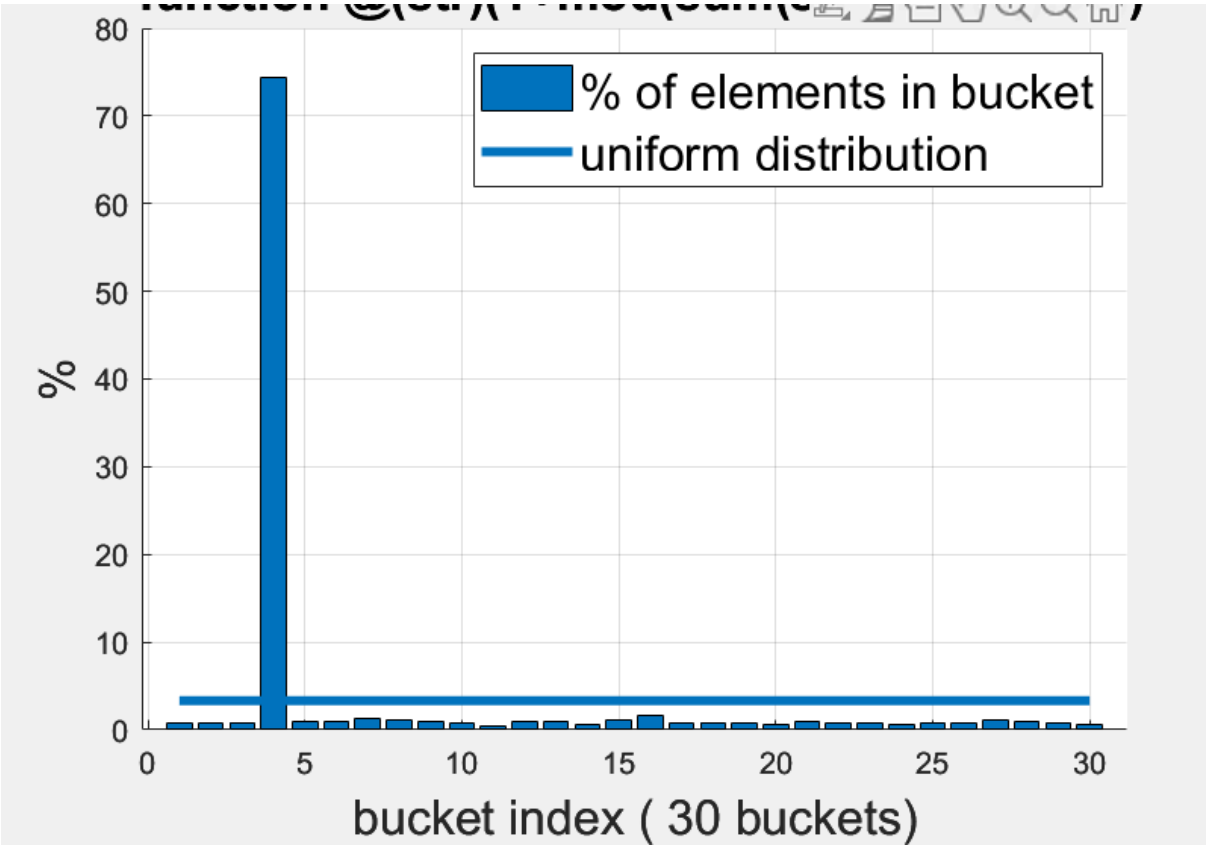
Não, pois para um número não primo, um conjunto não aleatório de dados pode criar um viés do resultado da função hash, por exemplo, o conjunto com muitos números divisíveis por 2, 3, 5, 6, 12 e 15(divisores de 30), criaria uma tabela de hash má distribuída.

2.2. Uma regra comum é usar um tamanho primo (e.g. 29) e não um tamanho com vários divisores, como 30. Que tipo de problema o tamanho primo evita, e porque a diferença não é muito grande no nosso exemplo?

Um conjunto não aleatório de dados pode criar um viés do resultado da função hash, por exemplo, o conjunto com muitos números divisíveis por 2, 3, 5, 6, 12 e 15(divisores de 30), criaria uma tabela de hash má distribuída.

2.3. Note que o arquivo mod30 foi feito para atacar um hash por divisão de tabela de tamanho 30. Explique como esse ataque funciona: o que o atacante deve saber sobre o código de hash table a ser atacado, e como deve ser elaborado o arquivo de dados para ataque.

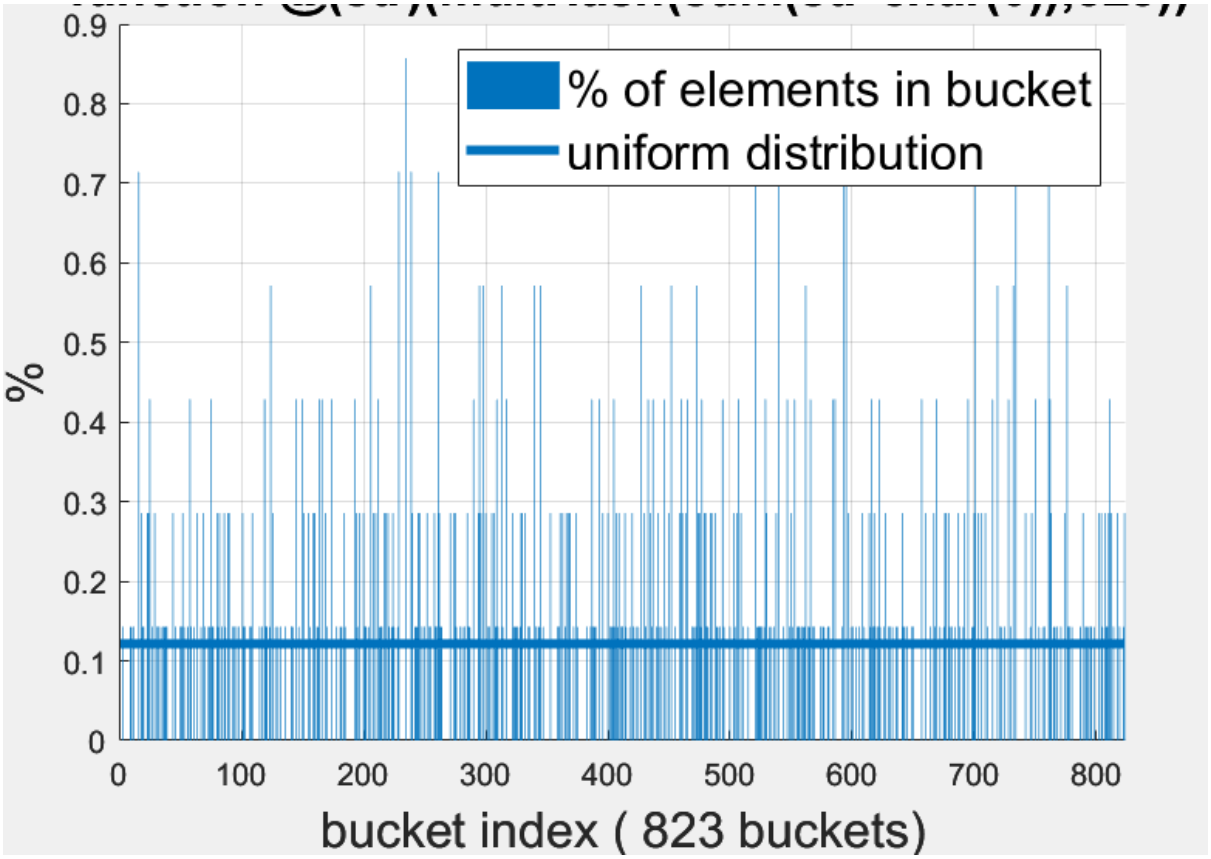
Assim, a tabela “mod30” foi montada de tal forma que o resto por 30 o ASCII de muitos dos seus valores seja 3. Ou seja, vai criar uma hash table de entropia muito baixa. No nosso exemplo acaba não sendo tão relevante pois estamos trabalhando com strings



3. Questão 3

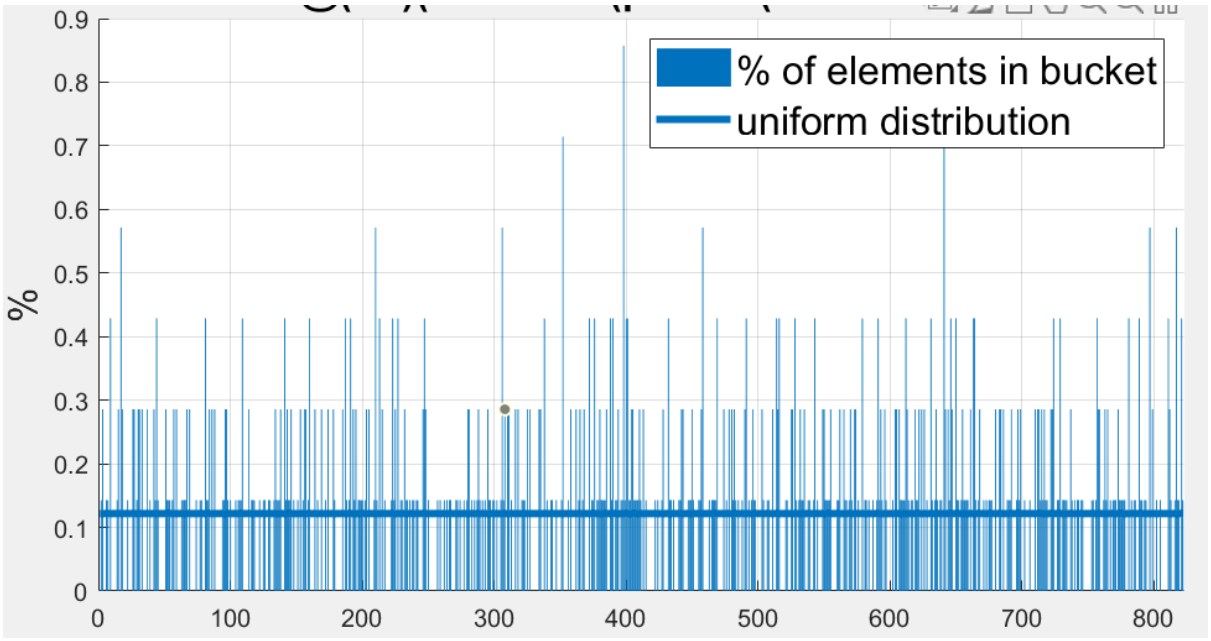
3.1. com tamanho 997 (primo) para a tabela de hash ao invés de 29, não deveria ser melhor? Afinal, temos 997 posições para espalhar números ao invés de 29. Porque às vezes o hash por divisão com 29 buckets apresenta uma tabela com distribuição mais próxima da uniforme do que com 997 buckets?

Um ocorre pois o caractere de menor ASCII ‘A’ possui valor 65, assim, escolhendo um hash grande demais(muito maior que 65) diminui a probabilidade de haver uma combinação linear de letras(palavra) tenha a soma resultante em de determinados valores, gerando um forte viés para um número pequeno de dados. Temos a seguinte distribuição para o conjunto “random”:

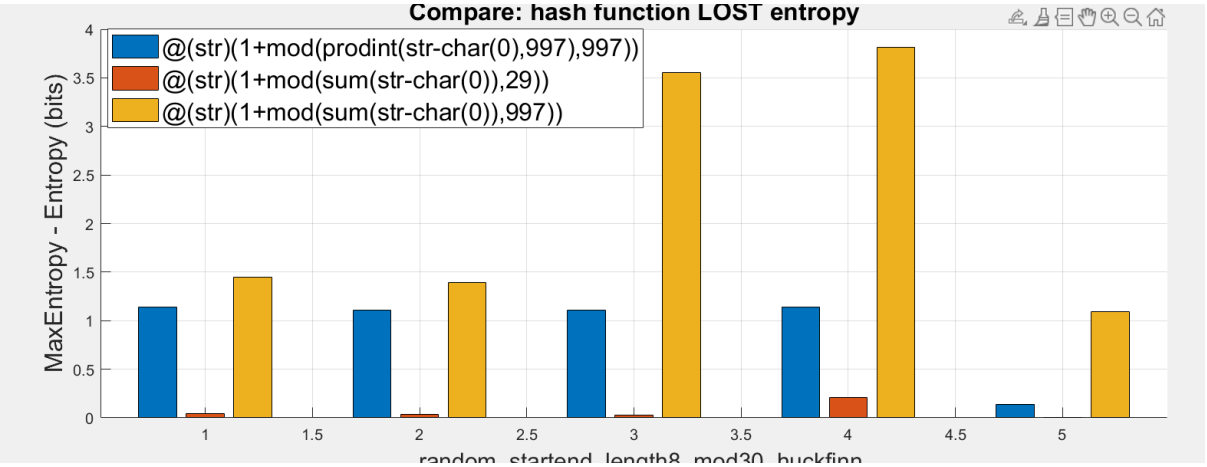


3.2. Porque a versão com produtório (prodint) é melhor?

Já para o produtório esse viés é gerado pela soma de números em um intervalo específico. Gerando um viés menor na distribuição dos dados, diminuindo a perda de entropia.



Assim, as perdas de entropia são:



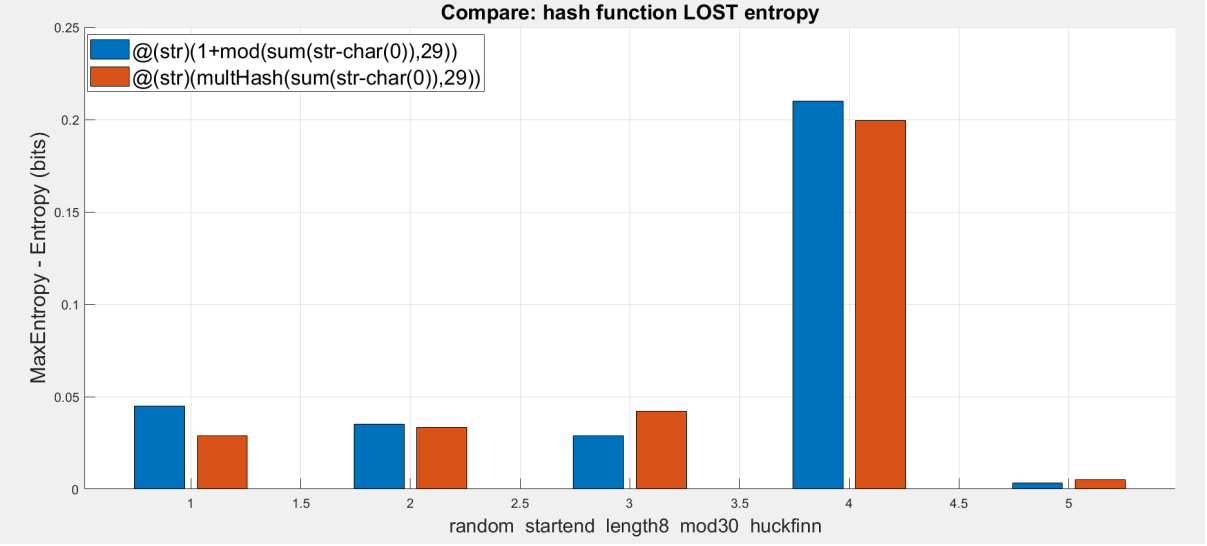
3.3. Porque este problema não apareceu quando usamos tamanho 29?

Para uma Hash Table menor esse problema é bastante atenuado por ser um tamanho menor que 65 e os intervalos com menor probabilidade acabam melhores distribuídos.

4. Questão 4

4.1. hash por divisão é o mais comum, mas outra alternativa é hash de multiplicação (NÃO É O MESMO QUE prodint.m, verifiquem no Corben). É uma alternativa viável? porque hashing por divisão é mais comum?

Sim, hash por multiplicação é uma alternativa viável que gera perda de entropia semelhante à da divisão.



No entanto, hash por multiplicação necessita maior poder computacional na maioria dos casos.

5. Questão 5

5.1. Qual a vantagem de Closed Hash sobre OpenHash, e quando escolheríamos Closed Hash ao invés de Open Hash? (pesquise! É suficiente um dos pontos mais importantes)

Uma vantagem do Closed Hashing é sua melhor performance em relação ao Open Hashing quando há um número limitado de entradas. Ou seja, o Closed Hashing poderia ser usado em um sistema de baixa rotatividade no qual a quantidade de valores a serem colocados na tabela é conhecida.

6. Questão 6

6.1. Suponha que um atacante conhece exatamente qual é a sua função de hash (o código é aberto e o atacante tem acesso total ao código), e pretende gerar dados especificamente para atacar o seu sistema (da mesma forma que o arquivo mod30 ataca a função de hash por divisão com tamanho 30). Como podemos implementar a nossa função de hash de forma a impedir este tipo de ataque? Pesquise e explique apenas a idéia básica em poucas linhas

Para evitar um ataque assumindo que o adversário conheça o código é possível determinar a função hash de maneira aleatória de maneira que não seja possível o atacante determinar qual função hash a qual foi escolhida.