جداول التقطيع

علينا تخزين بعض المعطيات من طبيعة ما(تسجيلات) والتي تقترن كل منها بقيمة خاصة تُسمى مفتاحKey والعمليات التي يجب تنفيذها هي ما يلي:

- إضافة تسجيلة جديدة
- حذف تسجيلة
- البحث عن تسجيلة اعتماداً على قيمة مفتاح

نريد طريقة فعالة وبسيطة لإجراء ذلك

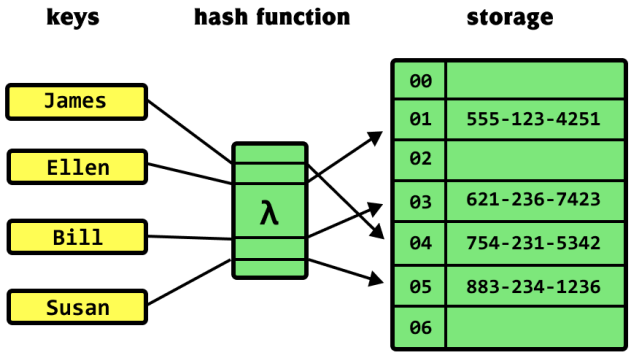
مفهوم جدول التقطيع :Hash Table

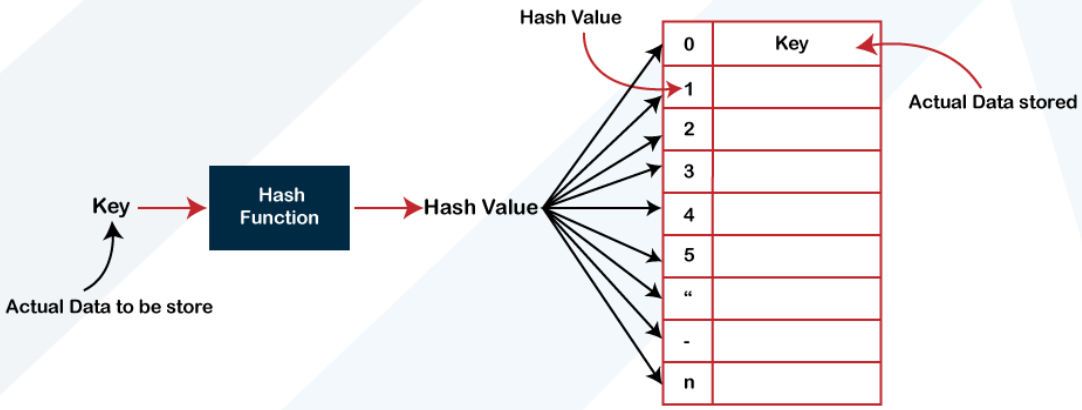
تعد من أنواع هياكل البيانات المهمة والسريعة وسهلة التطبيق، حيث يمثل البيانات على شكل أزواج (مفتاح وقيمة)، يتم تعيين كل مفتاح (key) لقيمة (value) في جدول التقطيع، حيث تُستخدم المفاتيح لفهرسة القيم/ البيانات، وتتم معالجة المفاتيح لإنتاج فهرس (index) جديد ليتم تعيينه للعنصر المطلوب، هذه العملية تسمى التقطيع .(hashing)

افرض أن لديك (object) وتريد تعيين مفتاح له لتسهيل البحث، ولتخزين زوج (المفتاح/ القيمة)، يمكنك استخدام المصفوفة البسيطة لهيكلة البيانات حيث أن المفاتيح هي (integers) ويمكن استخدام المؤشر (index)لتخزين القيم بشكل مباشر، مع ذلك، في الحالات التي تكون فيها المفاتيح كبيرة ولا يمكن استخدامها كمؤشر بشكل مباشر، يجب استخدام التقطيع (hashing)لهيكلة البيانات .

في التقطيع(hashing) ، يتم تحويل المفاتيح الكبيرة إلى مفاتيح صغيرة باستخدام دالة التقطيع، ومن ثم يتم تخزين القيم في جدول التقطيع، وفكرة التقطيع تقوم على توزيع المدخلات (زوج من المفتاح والقيمة) بشكل موحد عبر مصفوفة، ويتم تعيين مفتاح لكل عنصر (مفتاح محوّل). باستخدام هذا المفتاح، يمكنك الوصول إلى العنصر في وقت هو"O(1)" ، باستخدام المفتاح أيضاً، فإن الخوارزمية (دالة التقطيع) تقوم بحساب فهرس (index) والذي بدوره يشير لمكان يمكنك الاطلاع على قيمة أو إدراج قيمة .

من الأمثلة لجدول التقطيع في حياتنا، في الجامعات، يتم تخصيص رقم سجل فريد لكل طالب يُمكّنه من استخدامه لاسترداد المعلومات المتعلقة به، وفي المكتبات، يتم تخصيص رقم فريد لكل كتاب يمكن استخدامه لتحديد معلومات حول كتاب ما، مثل موقعه الدقيق في المكتبة أو المستخدمين الذين تم إصداره لهم وما إلى ذلك. نلاحظ أن في كلا المثالين، تم وضع رقم فريد للطلاب وللكتب .





تنفيذ جدول التقطيع:

حيث يتم تنفيذ التقطيع على خطوتين:

- يتم تحويل العنصر(value) ، إلى عدد صحيح باستخدام دالة تقطيع، ويمكن استخدام هذا العنصر كمؤشر لتخزين العنصر الأصلي الذي يقع في جدول التقطيع.
- يتم تخزين العنصر في جدول التقطيع، حيث يمكن استرجاعه بسرعة باستخدام مفتاح مجزأ.(hashed)

دالة التقطيع:Hash Function

دالة التقطيع هي أي دالة يمكن استخدامها لتعيين مجموعة بيانات ذات حجم كبير نسبيا لمجموعة بيانات ذات حجم ثابت، والتي تقع في جدول التقطيع، وتسمى القيم التي يتم إرجاعها بواسطة دالة التقطيع قيم التقطيع أو رموز التقطيع أو مجاميع التقطيع أو التقطيع ببساطة، لتحقيق هيكلة بيات باستخدام جدول التقطيع بشكل سليم، من المهم أن يكون لديك دالة تقطيع سليمة مع المتطلبات الأساسية التالية :

- **سهلة الحساب :**يجب أن تكون سهلة الحساب، ويجب ألا تصبح خوارزمية في حد ذاتها .
- **التوزيع الموحد :**يجب أن توفر الدالة توزيعًا موحدًا عبر جدول التقطيع، ويجب ألا يؤدي إلى تكتل .
- **تصادمات أقل :**تحدث التصادمات عند تعيين أزواج من العناصر إلى نفس قيمة التقطيع. (بغض النظر عن مدى جودة وظيفة التقطيع، لا بد أن تحدث التصادمات) للحفاظ على أداء جدول التقطيع، من المهم إدارة التصادمات من خلال تقنيات مختلفة لتحليل التصادم . أي تسمّى الحالة التي يرتبط فيها مفتاح مضاف حديثًا بموقع مشغول بعنصر آخر في جدول التطبيق بحالة التضارب ويجب التعامل معها والتخلص منها باستخدام بعض التقنيات الخاصة.

على سبيل المثال، تعدّ دالة التقطيع دالة سيئة إن أخذت أول ثلاثة أعداد في رقم الهاتف، ولكن يمكن تحسين هذه الدالة بجعلها تأخذ آخر ثلاثة أعداد من رقم الهاتف، وهذا لا يعني أنّ هذه الدالة هي أفضل دالة تقطيع في هذا الصدد، بل يمكن اتباع طرق أخرى لجعلها دالة أفضل .

تكون دالةُ `h()` دالةَ تقطيع Hash Functions إذا كانت تأخذ عنصرًا `x ∈ X` من أيّ حجم، وتعيد قيمة `y ∈ Y` من حجمٍ ثابت .y = h (x).

تتميز دوّال التقطيع النموذجية بالخصائص التالية :

- تتصرف مثل توزيعات منتظمة uniform distribution
- دوال التقطيع حتميةdeterministic ، حيث ينبغي أن تعيد الدالة `h(x)` القيمة نفسها دائمًا لعنصر `x` محدد .
- ينبغي أن تكون سريعة الحساب) ذات تعقيد زمني .(O (1)).

حجم قيمة التقطيع عمومًا أصغر من حجم البيانات المُدخلة `|y| < |x|` :، علاوة على أنّ دوال التقطيع غير قابلة للعكس، لأنه يجوز أن تكون لقيمتين مختلفتين قيمة التقطيع نفسها، أو بتعبير رياضي :

```
∃ x1, x2 ∈ X, x1 ≠ x2: h(x1) = h(x2)
```

قد تكون المجموعة `X` منتهية أو غير منتهية، أما `Y` فهي دائمًا مجموعة منتهية. وتُستخدم دوال التقطيع في الكثير من مجالات الحوسبة مثل هندسة البرمجيات والتشفير وقواعد البيانات والشبكات وتعلُّم الآلة ونحو ذلك، وهناك العديد من أنواع دوال التقطيع، ولكل منها خصائص مميزة .

تكون قيمة التقطيع عددًا صحيحًا في الغالب، ومعظم لغات البرمجة لديها توابع ودوال خاصة لحساب قيم التقطيع، مثل الدالة `GetHashCode()` في لغة #C التي تعيد قيمة من النوع) `Int32` عدد صحيح من 32 بت) لكل الأنواع. وكذلك في لغة جافا هناك تابع `hashCode()` يحسب قيمة التقطيع -من النوع int- لكل صنف من أصناف جافا .

تطبيقات على جدول التقطيع:Hash Table

- **فهرسة قاعدة البيانات :**يمكن أيضًا استخدام جداول التقطيع كتركيبات بيانات(disk-based) ، ومؤشرات قاعدة بيانات.
- **ذاكرات التخزين المؤقت :(Caches)** يمكن استخدام جداول التقطيع لتنفيذ (implement) ذاكرات التخزين المؤقت، على سبيل المثال، جداول البيانات المساعدة التي تُستخدم لتسريع الوصول إلى البيانات، والتي يتم تخزينها بشكل أساسي في وسائط أبطأ.

- **تمثيل الـ (object):** تستخدم العديد من اللغات الديناميكية، مثل Perl) و Python وJavaScript وRuby)، جداول التقطيع لتنفيذ الـ(objects).
- تُستخدم دوال التقطيع في العديد من الخوارزميات لجعل الحوسبة أسرع.

طرق تعريف دوال التقطيع Hash methods

هناك عدة طرق لتعريف دوال التقطيع، يمكننا أن نفترض دون الإخلال بالعمومية، أنّ المجموعة Xتساوي مجموعة الأعداد الصحيحة الموجبة، و xعنصر منها: أي $x \in X = \{z \in \mathbb{Z}: z \geq 0\}$. وليكن m عددًا، ويُفضّل أن يكون أوليًا (شرط أن لا تكون قيمته قريبة جدًا من أي قوّة للعدد 2) .

فيما يلي طريقتان لحساب قيم التقطيع الخاصة بعناصر X:

الطريقة	دالة التقطيع
طريقة القسمة	$h(x) = x \bmod m$
طريقة الضرب	$h(x) = \lfloor m\ (xA \bmod 1) \rfloor,\ A \in \{z \in \mathbb{R}: 0 < z < 1\}$

جداول التقطيع

تُستخدم دوال التقطيع مع جداول التقطيع hash tables لحساب الفهارس في مصفوفة من الحجراتarray of slots ، وجدول التقطيع هي هيكلية بيانات لتنفيذ **القواميس** dictionariesوهي بيانات من نوع مفتاح-قيمة .

التعقيد الزمني لتطبيقات لجداول التقطيع يساوي في العادة O (1) للعمليات التالية :

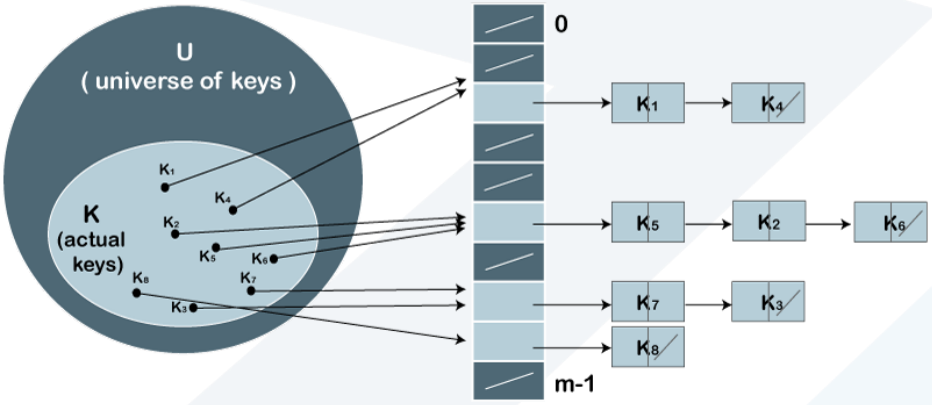
- إدراج البيانات بحسب قيمة المفتاح .
- حذف البيانات بحسب قيمة المفتاح .

يمكن أن يكون لعدة مفاتيح نفس التقطيع (الحجرة)، وعندئذ فهناك طريقتان لحل هذه المشكلة :

- **السلسلة Chaining**: تُستخدم القوائم المترابطة لتخزين العناصر التي لها نفس قيمة التقطيع في الحجرة .

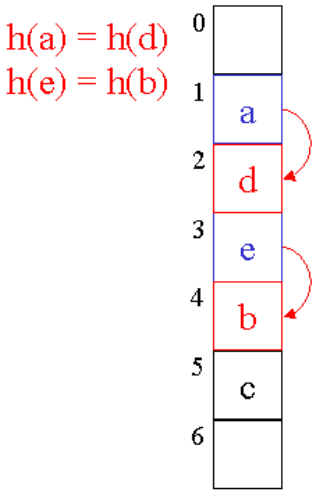
  في هذه الطريقة تشير كل خلية في جدول التقطيع إلى قائمة مترابطة تحتوي على سجلات تمتلك نفس قيمة التقطيع .

Collision Resolution by Chaining



- **العنونة المفتوحة: Open addressing** يُخزّن عنصر واحد على الأكثر في كل حجرة .

تخزّن جميع العناصر في هذه الطريقة في جدول التقطيع نفسه؛ لذا يلزم أن يكون حجم الجدول أكبر من العدد الكلي للمفاتيح (أو يكون مساويًا له (لاحظ أنّ بالإمكان زيادة حجم الجدول عن طريق نسخ البيانات القديمة إن اقتضت الحاجة.



تتضمن طريقة العنونة المفتوحة العمليات التالية :

- `Insert(k)`: يجري سبر خلايا جدول التقطيع إلى حين الوصول إلى خلية فارغة، ويدرج العنصر `k` فيها.
- `Search(k)`: يجري سبر خلايا جدول التقطيع إلى أن لا يتساوى مفتاح الخلية مع قيمة `k` أو عند الوصول إلى خلية فارغة.
- `Delete(k)`: لا يُحذف المفتاح مباشرة لأنّ ذلك يؤدي إلى إيقاف عملية البحث؛ لذا تعلّم الخلايا المحذوفة بالعلامة `deleted`. يمكن للعملية `Insert` أن يدرج عنصرًا في الخلية المحذوفة، ولكن عملية البحث لن تتوقف عند الوصول إلى مثل هذه الخلايا.
-

تُستخدم الطرق التالية لحساب تسلسلات المسبار probe sequences المطلوبة في العنونة المفتوحة :

الطريقة	الصيغة
السبر الخطي	h(x, i) = (h"(x) + i) mod m
السبر التربيعي	h(x, i) = (h"(x) + i^2) mod m
السبر المضاعف	h(x, i) = (h1(x) + i*h2(x)) mod m

الدوال h"(x) وh1(x) وh2(x) هي دوال مساعدة، و i ينتمي إلى المجموعة {0, 1, ..., m-1}

1. السبر الخطي: Linear Probing

يُسبر جدول التقطيع في هذه الطريقة خطّيًا بحثًا عن الخلية التالية. وعادة ما يكون الفاصل بين سبرين 1 كما هو الحال في المثال أدناه .

لنفترض أنّ hash(x) هي موقع الخلية الذي تم حسابه باستخدام دالة التقطيع وأنّ S هو حجم الجدول .

إن كانت الخلية hash(x) % S ممتلئة، سنحاول إجراء (hash(x) + 1) % S.

إن كانت الخلية (hash(x) + 1) % S ممتلئة، سنحاول إجراء (hash(x) + 2) % S.

إن كانت الخلية (hash(x) + 2) % S ممتلئة، سنحاول إجراء (hash(x) + 3) % S.

إن أكبر مشكلة تواجه عملية السبر الخطي هي حالة التجمّع clustering ، حيث تشكّل العناصر المتتابعة مجموعات تؤدي إلى زيادة الوقت الذي يستغرقه إيجاد خلية فارغة أو البحث عن عنصر معيّن .

2. السبر التربيعي Quadratic Probing

نبحث في هذه الطريقة عن الخلية i^2 في الدورة ذات الرقم i.

لنفترض أنّ hash(x) هي موقع الخلية الذي تم حسابه باستخدام دالة التقطيع وأنّ S هو حجم الجدول .

إن كانت الخلية hash(x) % S ممتلئة، سنحاول إجراء (hash(x) + 1*1) % S.

إن كانت الخلية (hash(x) + 1*1) % S ممتلئة، سنحاول إجراء (hash(x) + 2*2) % S.

إن كانت الخلية (hash(x) + 2*2) % S ممتلئة، سنحاول إجراء (hash(x) + 3*3) % S.

3.التقطيع المضاعفDouble Hashing

نستخدم في هذه الطريقة دالة تقطيع أخرى hash2(x) ونبحث عن i*hash2(x) في الدورة i.

لنفترض أنّ hash(x) هي موقع الخلية الذي تم حسابه باستخدام دالة التقطيع وأنّ S هو حجم الجدول .

إن كانت الخلية hash(x) % S ممتلئة، سنحاول إجراء

(hash(x) + 1*hash2(x)) % S.

إن كانت الخلية (hash(x) + 1*hash2(x)) % S ممتلئة، سنحاول إجراء

(hash(x) + 2*hash2(x)) % S.

إن كانت الخلية (hash(x) + 2*hash2(x)) % S ممتلئة، سنحاول إجراء

(hash(x) + 3*hash2(x)) % S.

المقارنة بين العنونة المفتوحة والسلاسل المنفصلة

يعقد الجدول التالي مقارنة بين طريقة العنونة المفتوحة والسلاسل المنفصلة :

	العنونة المفتوحة	السلاسل المنفصلة
1	تحتاج هذه الطريقة إلى إجراء حسابات مطولة.	هذه الطريقة أسهل في التنفيذ.
2	يمكن لجدول التقطيع أن يمتلئ.	لا يمتلئ جدول التقطيع في هذه الطريقة إطلاقًا، ويمكننا إدراج المزيد من العناصر في السلسلة.
3	تتطلب هذه الطريقة المزيد من العناية وذلك لتجنب الوقوع في مشكلة التجمّع ومعامل التحميل.	هذه الطريقة أقل تحسّسًا لدالة التقطيع أو معامل التحميل.
4	تستخدم هذه الطريقة عندما يكون عدد المفاتيح وتواترها معلومًا.	تستخدم هذه الطريقة غالبًا عندما لا يكون عدد وتواتر المفاتيح المراد إضافتها أو حذفها معلومًا.
5	تقدّم هذه الطريقة أداءً أفضل مع الذاكرة المخبئية وذلك لأنّ كل شيء يُخزّن في جدول التقطيع.	أداء هذه الطريقة مع الذاكرة المخبئية ليس جيّدًا وذلك لأنّ المفاتيح تخزّن في قوائم مترابطة.
6	يمكن استخدام الخلية في هذه الطريقة حتى إن يرتبط المدخل بتلك الخلية.	هدر في المساحة، إذ لا يُستخدم جدول التقطيع بأكمله.
7	لا توجد روابط في هذه الطريقة.	تستخدم هذه الطريقة مساحة إضافية لأجل الروابط.

يعرض المثال التالي كيفية معالجة حالة التضارب بواسطة طريقة السلسلة المنفصلة في لغة :C++

```
#include<iostream>
#include <list>
using namespace std;

class Hash
{
        int BUCKET; // No. of buckets

        // مؤشر إلى مصفوفة
    // Pointer to an array containing buckets
        list<int> *table;
public:
        Hash(int V); // دالة بانية

        // تدرج هذه الدالة مفتاحًا في جدول التقطيع
        void insertItem(int x);

        // تحذف هذه الدالة مفتاحًا من جدول التقطيع
        void deleteItem(int key);

        // دالة تقطيع تربط القيم بالمفاتيح
        int hashFunction(int x) {
                return (x % BUCKET);
        }

        void displayHash();
};

Hash::Hash(int b)
{
        this->BUCKET = b;
        table = new list<int>[BUCKET];
}

void Hash::insertItem(int key)
{
        int index = hashFunction(key);
        table[index].push_back(key);
}

void Hash::deleteItem(int key)
{
// الحصول على موقع المفتاح في جدول التقطيع
int index = hashFunction(key);

// الحصول على المفتاح في الموقع المعطى ضمن المصفوفة
list <int> :: iterator i;
for (i = table[index].begin();
                i != table[index].end(); i++) {
        if (*i == key)
        break;
}

// إن كان المفتاح موجودًا في جدول التقطيع فسنحذفه
if (i != table[index].end())
```

```
        table[index].erase(i);
}

// دالة لعرض جدول التقطيع
void Hash::displayHash() {
for (int i = 0; i < BUCKET; i++) {
        cout << i;
        for (auto x : table[i])
        cout << " --> " << x;
        cout << endl;
}
}

// اختبار الدوال السابقة
int main()
{
// مصفوفة تتضمن المفاتيح التي ستضاف إلى الجدول
int a[] = {15, 11, 27, 8, 12};
int n = sizeof(a)/sizeof(a[0]);

// تدرج الدالة المفاتيح في جدول التقطيع
Hash h(7);
    // عدد العناصر في جدول التقطيع هو 7
for (int i = 0; i < n; i++)
        h.insertItem(a[i]);

// تحذف الدالة العنصر 12 من جدول التقطيع
h.deleteItem(12);

// تعرض الدالة جدول التقطيع
h.displayHash();

return 0;
}
```

تعطي الشيفرة السابقة المخرجات التالية :

```
0
1 --> 15 --> 8
2
3
4 --> 11
5
6 --> 27
```