31. **Геш-функція**[[1]](https://uk.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F" \l "cite_note-1) (**Хеш-функція)** — [функція](https://uk.wikipedia.org/wiki/%D0%9F%D1%96%D0%B4%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0" \o "Підпрограма), що перетворює вхідні дані будь-якого (як правило великого) розміру в дані фіксованого розміру.

**Гешування** (хешування, [англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0" \o "Англійська мова) *hashing*) — перетворення вхідного масиву даних довільної довжини у вихідний бітовий рядок фіксованої довжини. Такі перетворення також називаються **геш-функціями** або **функціями згортання**, а їхні результати називають гешем, **геш-кодом**, **геш-сумою**, або **дайджестом повідомлення**

Геш-функція використовується зокрема у структурах даних — [геш-таблицях](https://uk.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D1%8F), широко вживаних у програмному забезпеченні для швидкого пошуку даних. Геш-функції використовуються для оптимізації таблиць та баз даних за рахунок того, що у однакових записів однакові значення геш-функції. Такий підхід пошуку дублікатів ефективний у файлах великого розміру. Прикладом цього буде знаходження подібних ділянок у послідовностях [ДНК](https://uk.wikipedia.org/wiki/%D0%94%D0%9D%D0%9A). [Криптографічна геш-функція](https://uk.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D1%96%D1%87%D0%BD%D0%B0_%D0%B3%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F) дозволяє легко перевірити, що деякі вхідні дані зіставляються із заданим значенням гешу, але, якщо вхідні дані невідомі, то навмисно важко відновити вхідне значення (або еквівалентну альтернативу), знаючи збережене значення геш-функції. Це використовується для забезпечення [цілісності](https://uk.wikipedia.org/wiki/%D0%A6%D1%96%D0%BB%D1%96%D1%81%D0%BD%D1%96%D1%81%D1%82%D1%8C_%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D1%97" \o "Цілісність інформації) переданих даних, і є будівельним блоком для [HMACs](https://uk.wikipedia.org/wiki/HMAC), які забезпечують [аутентифікацію повідомлень](https://uk.wikipedia.org/wiki/%D0%90%D1%83%D1%82%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D1%96%D0%BA%D0%B0%D1%86%D1%96%D1%8F_%D0%BF%D0%BE%D0%B2%D1%96%D0%B4%D0%BE%D0%BC%D0%BB%D0%B5%D0%BD%D1%8C" \o "Аутентифікація повідомлень).

Геш-функції пов'язані (і їх часто плутають) з [контрольною сумою](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8C%D0%BD%D0%B0_%D1%81%D1%83%D0%BC%D0%B0" \o "Контрольна сума), контрольними цифрами, [відбитками пальців](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D0%B4%D0%B1%D0%B8%D1%82%D0%BA%D0%B8_%D0%BF%D0%B0%D0%BB%D1%8C%D1%86%D1%96%D0%B2_(%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)" \o "Відбитки пальців (інформатика)), [рандомізацією функцій](https://uk.wikipedia.org/wiki/%D0%A0%D0%B0%D0%BD%D0%B4%D0%BE%D0%BC%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%97" \o "Рандомізація функції), кодами, що виправляють помилки, і з шифрами. Хоча ці поняття певною мірою збігаються, кожен з них має свою власну область застосування і вимоги і є розробленим і оптимізованим по-різному

Гешування застосовується для побудови асоціативних масивів, пошуку дублікатів в серіях наборів даних, побудови унікальних ідентифікаторів для наборів даних, контрольного підсумовування з метою виявлення випадкових або навмисних помилок при зберіганні або передачі, для зберігання паролів в системах захисту (у цьому випадку доступ до області пам'яті, де знаходяться паролі, не дозволяє відновити сам пароль), при виробленні електронного підпису (на практиці часто підписується не саме повідомлення, а його геш-образ).

У загальному випадку однозначної відповідності між вихідними даними і геш-кодом немає в силу того, що кількість значень геш-функцій менша, ніж число варіантів значень вхідного масиву. Існує безліч масивів з різним вмістом, що дають однакові геш-коди — так звані колізії. Імовірність виникнення колізій відіграє важливу роль в оцінці якості геш-функцій.

Розроблено багато алгоритмів гешування з різними властивостями (розрядність, обчислювальна складність, криптостійкість тощо). Вибір тієї чи іншої геш-функції визначається специфікою розв'язуваної задачі. Найпростішими прикладами геш-функцій можуть служити контрольна сума або CRC.

33.**Граф** — це сукупність об'єктів із зв'язками між ними.

Якщо ребро з'єднує дві вершини, то кажуть, що воно***інцидентне*** цим вершинам, а вершини, які з'єднані таким ребром, назива­ються***суміжними****.* Якщо кінці ребра належать одній вершині, то таке ребро називається***петлею****.*

Вершини, які не належать кінцям жодного з ребер у графі, на­зиваються***ізольованими****.*

*Степенем вершини* будемо називати число ребер, яким нале­жить ця вершина. Або ще можна сказати, що степенем вершини називається кількість ребер, інцидентних даній вершині. Позначається степінь вершини*а* як*Р(а).*

У свою чергу кажуть, що вершини*а1* та*а -* є***зв'язними****,* якщо існує шлях від *аі* до*а-.*

Надалі будемо називати граф***зв'язним****,* якщо будь-яка пара його вершин зв'язна. Зрозуміло, що повний граф завжди є зв'язним, але не всякий зв'язний граф є повним.

Введемо ще один термін:***циклом*** називається шлях, в якому збігаються початкова і кінцева вершини.

Якщо цикл через кожну вершину проходить лише один раз, то такий цикл називається***простим****.*

***Довжиною шляху*** (циклу) називається кількість ребер цьо­го шляху (циклу).

Для неорієнтованого ребра графа порядок, в якому вказую­ться вершини, які воно з'єднує, неважливий. Для орієнтова­ного ребра першою вказується вершина, з якої воно виходить. В орграфі також існує поняття орієнтованого шляху: це послі­довність ребер між двома вершинами з урахуванням їхньої орієнтації. Шлях в орграфі, який є циклом, називається орієн­тованим циклом.

Для орієнтованих графів специфічним є визначення степе­нів вершин. Для кожної з них окремо визначається***вхідний******степінь****,* що дорівнює кількості ребер, які входять у вершину, і***вихідний******степінь****,* що визначається кількістю ребер, які ви­ходять із неї. Сума вхідного і вихідного степенів вершини орієнтованого графа називається степенем цієї вершини.

Якщо ж у графі вказана ще й «вага» кожного ребра, то та­кий граф називається***зваженим****.* Тобто існують неорієнтовані зважені графи та орієнтовані зважені графи

граф називається***ейлеровим****,* якщо у ньому можна повер­нутися в ту саму вершину, з якої ми вийшли, обійшовши кожне з ребер тільки один раз. Існують також відповідно поняття ейлерових шляхів та циклів у графі.

Граф називається***гамільтоновим****,* якщо у ньому можна по­вернутися в ту саму вершину, з якої ми вийшли, обійшовши кожну з вершин тільки один раз. Зрозуміло, що можна говори­ти про існування гамільтонових шляхів та циклів у графі

#### 34. 1. Обхід графа

**Ідея**

Обхід починається з вершини *v*. Під час обходу вершини, суміжні з поточною, відмічаються і вносяться до структури даних *T*(стеку або черги). Чергова вершина для перегляду береться зі структури *T*, поки *T* не стане порожньою. Якщо як структура даних використовується стек, то говорять про *пошук у глибину*, якщо черга — про *пошук у ширину*.

Для відмічання вершин використовується масив *marks*, початкові значення елементів якого дорівнюють 0 (вершини невідмічені).

**Псевдокод**

Вставити(*v*, *T*)  
*marks*[*v*]=1   
**поки***Т*непорожня{ *u* =Вилучити(*T*)  
  Вивести *u*  
**Для**всіх*t*з\_множини\_вершин*V*  
    **якщо** *t* суміжна\_з *u* **і** *marks*[*t*]==0  
      **то** {Вставити(*t*, *T*); *marks*[*t*]=1}  
**}**

**2.Пошук найкоротших шляхів**

**Ідея**

Поступово уточнювати шукані шляхи:

* спочатку покласти найкоротші шляхи рівними довжині ребра від джерела (вершини 1) до відповідної вершини;
* на кожній ітерації намагатися зменшити шлях, замінивши "прямий" щлях "обхідним" через вершину, яка знаходиться на найменшій віддалі.

Алгоритм використовує множину *S*вершин, для яких найкоротші шляхи відомі та масив *d*цих найкоротших шляхів.

**Псевдокод**

// **ініціалізація**  
S={1}   
**для**  *i*  **від**2  **до**  *n*   
  *d*[*i*]=довжинаребра(1, *i*)  
// **у**точнення\_шляхів  
**для**  *i*  **від** 2  **до**  *n* –1  
{   
  вибрати з *V* \ *S* вершину *w*, для якої *d*[*w*] мінімально   
  *S* = *S* +{*w*};   
  **для** кожної *v* з *V* \ *S*   
    *d*[*v*]= min{*d*[*v*],*d*[*w*]+довжина ребра (*w*,*v*)}  
}

35.

* **матриця суміжності**: елемент з індексами *i*, *j* дорівнює 1, якщо вершини *i* та *j* є суміжними, і 0 в іншому випадку;
* **матриця інциденцій**: елемент з індексами *i*, *j* дорівнює 0, якщо вершина *i*не є індидентною ребру *j*; 1, якщо інцидентна (для орграфа 1 - якщо вершина *i*є кінцем ребра *j*, і -1, якщо початком);
* **масив ребер (дуг)**: масив, кожен елемент якого є пара суміжних вершин;
* **список суміжності:**масив, кожен елемент якого є покажчиком на список суміжних вершин.