Хеширане.
Ако приемем че имаме едно множество от информационни съобщения, в което ние трябва да намерим точно определено съобщение. За да не прочитаме цялото съобщение, се изгражда ключ за всяко съобщение. При търсене на определено съобщение ние ще търсим неговия ключ. Ако M = (A,B,C, .. Z) е нашето множество от информационни съобщения и ако A = (1,2, 3, ... 25) са нашите ключове и тяхното разположение според адреси в един масив, то тогава функцията която дава връзката между дадено съобщение и адреса на неговия ключ е следната: h (M) = A където h е функцията (hash – раздробяване).

Пример за свързване на съобщения с позициите на техните ключове може да бъде:
h(a)=0 ; h(B)=1 ; …. h(Z)=25.

H – функцията е функция обратна на h. Ако h(D) = 3 тогава H(3) = D.
H(0) = A ; H(1) = B ; H(25) = Z

Хеширането представлява метод, с който се изграждат ключовете, метод по който те се подреждат в таблица (масив) и начин за намиране на стойност, която вече е в масива.

Имаме 6 записа от X1 до X6 всеки от тях съдържа определена 8 битова стойност:
X1 = 1010 1 111 X2 = 0110 0 110 X3 = 0110 0 111
X4 = 0011 0 100 X5 = 1000 0 001 X6 = 1011 1 000

Също така имаме маска, според маската взимаме определени стойности от записите, а останалите пропускаме. В случая взимаме последните 3 символа от записите.
Маска = 0000 0 111 - Взимат се стойностите от xi които в маската са с 1-ца
Чрез хеширане се получават следните стойности от 0 до 7, диапазона (0-7) се определя от броя на битовете който имат значение при хеширането. В нашия случай 3 бита определят 8 стойности, които сме избрали да са от 0 до 7. Със следната последователност получаваме стойността която се явява позиция в която трябва да запишем ключа в масива.

Хеширане от (начално съобщение) -> хеширано съобщение -> стойност.
h(X1) -> 101 ->5 h(X4) -> 100 -> 4
h(X2) -> 110 -> 6 h(X5) -> 001 -> 1
h(X3) -> 111 -> 7 h(X6) -> 000 -> 0

в масива от ключове към всяка позиция записваме съответния ключ и на коя стойност той принадлежи.

A – позиции	H - стойности
0	x6
1	x5
2	-
3	-
4	x4
5	x1
6	x2
7	x3

Възможно е получаването на еднакви хеширани стойности от различни входни данни. В този случай резултатите съвпадат и различни данни трябва да се поставят на едно място в таблицата. Това е неправилно и решението на проблемът е повтарящите се данни да се поставят на друго свободно място. Този проблем се нарича колизия.

Пример:	Получени ключове:	А - стойности	H
X1 = 1010 1 101	h(X1) = 111 = 7	0	x3
X2 = 0110 0 110	h(X2) = 110 = 6	1	x5
X3 = 0110 0 101	h(X3) = 111 = 7	2	-
X4 = 0011 0 100	h(X4) = 100 = 4	3	-
X5 = 1000 0 001	h(X5) = 001 = 1	4	x4
X6 = 1011 1 100	h(X6) = 000 = 4	5	x6
		6	x2
Маска: 0000 0 111		7	x1

Тук x3 е в колизия с x1, x6 е в колизия с x4. За поставяне на елементите на колизията е избрано следващото свободно място, но това не е добро решение, защото е голяма вероятността това също да предизвика колизия. Друг вариант е поставянето на произволно място, което също не дава гаранция, че няма да се предизвика отново колизия. Възможни са два вида колизия. Пряка когато елементи със същата стойност е трябвало да се поставят на едно място. Непряка когато елемент преминал през колизия е поставен на свободно не съответстващо на стойността му място отново поражда колизия с елемент който трябва да бъде поставен на точно това място в последствие.

„Преобразуване от име в адрес.” - Ако имаме произволно съобщение и искаме да го хешираме имаме 2 случая. Ако съобщението е малко не е необходимо да се взима част от съобщението за да се направи неговия ключ. Ако съобщението е голямо, първо ще изберем коя с част от съобщението да направим нашия ключ.

Идентификатор (ключ) $K = d_{n-1} * W^{n-1} + d_{n-2} * W^{n-2} + ... d_1 * W^1 + d_0 * W^0$
Където W е видът на бройната система.

Пример W = 26
AEG= $0.26^2 + 4.26^1 + 6.26^0 = 110$
CPU = $2.26^2 + 15.26^1 + 20.26^0 = 1762$
Ако имаме по дълъг израз можем да изберем различни части от него за изграждане на ключа.
procesor -> pro | ced | ure
pro = $15.26^2 + 17 . 26^1 + 14 . 26^0 = 10596$
ced = $2.26^2 + 4.26^1 + 3.26^0 = 1459$
ure = $20.26^2 + 17.26^1 + 4.26^0 = 13966$
10596 | 1459 | 13966 – различни стойности получени от различните избрани ключове.

Избор на h функция. Използва се когато искаме да разположим стойностите по специален начин в таблицата (масива), а не просто те да се поставят на място със пореден номер равно на ключът на въпросните стойности. Има различни подходи. С умножение, с деление, със сума по модул от 2, както и много други.
А) С деление. $H(V) = V \bmod H+B$ Където H - големи на таблицата; B - отместване (при колизия). Броят на полетата в таблицата не трябва да бъде четно число.
Трябва да използваме прости числа за броя на полетата в таблицата.
Б) С умножение.
$f(V) = C.V \quad 0<C<1 \quad C = Ф - 1$ Където Ф е такова число, че $1/Ф = Ф – 1$. Тогава $C = 0,618$.
$h(V) = H.f(V) \quad H = 2^k$ където „к” може да се вземе като стойност от $\frac{B}{K}$ където „к” може да бъде броя на ключовите елементи от маската, а „в” останалите елементи.

Как да подберем маската възможни са много подходи. И изборът е оставен на нас. Но трябва да се стремим избраният от нас подход да извлича тези стойности от съобщенията който имат най голяма вероятност за различния. За да не се получи вариант в който от голям брой различни съобщения ние сме избрали повтаряща се техен сегмент и сме получили огромен брой различни ключове.
Метод на средата на квадрат. $H = 2^k$ В този метод подбираме така стойностите, че взимаме средната стойност от записа, след което средните спрямо нея от ляво и от дясно и тн.

3. Обработка на колизиите.
А) Метод с вътрешна адресация.
Б) Метод с използване на отделна област на препълване.
За първите 2 метода можем да подходим по различни начини. Правен на таблица, в която да записваме, кой елементи са в колизия и на кое място е трябвало да бъдат по правило и на кое място са записани след участие в колизията. Отделяне на допълнително памет за разполагане на елементи участвали в колизия. Тези както и други подходи се взимат изцяло от поставените задачи и нашето решение.

Пробинг – начин за търсене на случани клетки
1 – линеен (още нормален пробинг) Проверява следващия елемент.
Fi(k) = (h(k)+ gi) mod H gi = a . I gi(k)=a(k).i I = 0,1,2 … - nномер на пробата

В този метод след като е установено че мястото в което трябва да запишем един елемент е зает от друг и следователно имаме колизия, то тогава обхождаме последователно останалите докато не намерим свободно място. Този метод е неодобен, довежда до струпване на елементи. Предпочитат се някой от другите методи, като при тях се цели разпространение на елементите равномерно в таблицата.

2 – квадратичен gi = a . i + b . I^2 gi(k) = a.i+b(k).i a(k)=b(k) = V div H
3 – случаен -псевдо случайни числа. Избират се случайни полета за записване на стойностите. Компютърните системи работят по алгоритми написани от някого другиго и липсата на собствен интелект ги лишава от възможността да изберат случайни величини. Този фактор ги карад а работят с алгоритми за случайни величини като те обаче действат по някаква последователност и често повтарят едни и същи решения, което противоречи на чистата случайност.
а) R- псевдо
$Rn = (a R_{n-1} + b)$
$R_0 = h(k)$

В) списъчни колизии
Всеки елемент който се явява в колизия се на място в паметта до което се достига с оказател от предходния елемент участващ в същата колизия.

[ID|C|D|L|*|данни или указател]
ID-идентификационен номер
D – флаг за изтриване
L – флаг за връзка
P_0 – указател към областта на препълване

За намиране на минималното скелетно (обхващане) дърво на графа има различни начини. Тук ще разгледаме три от тях.

1 Метод на „Крускал”
Използва се таблица съдържаща теглото на всяко ребро, двата възела, между който се намира и списъци за вече свързаните възли. Като таблицата е подредена според тегловите коефициенти (стойностите на дъгите) във възходящ ред. Обхождайки таблицата ред по ред използваме само дъги който свързват отделни списъци който не са били свързани по никакъв начин преди това и така се изгражда нов списък.
За N-брой ребра алгоритъмът има сложност O = N.logN при основа 2. Тази сложност се получава от необходимостта да подредим ребрата в нарастващ ред според тегловите им коефоциенти.

По време на упражнение, се прави произволен граф и таблица, с които се демонстрира метода.

2 Метод на „Прим” при който се свързват стойности само ако свързването става между несвързана стойност и която и да е стойност но от вече съществуващата поредица (списък). Като тук се използва само 1 поредица (тоест не се изграждат отделни списъци както в предишния метод).

По време на упражнение, се прави произволен граф с който се демонстрира метода.

3 Метод на “Dikstra”. Този метод е много сходен с предишния но при него за определяне на тегловия коефициент на всяко ребро се взимат в предвид и тегловите коефициенти на ребрата които ползваме за да стигнем до въпросното, като започнем от възела от, който сме започнали обхождането.

По време на упражнение, се прави произволен граф с който се демонстрира метода.