A. Caz rau = caz mediu = caz defav => Timpii mediu si defav. sunt egali

Notăm $n = 2^k \Rightarrow k = \log_2 n$

$T(2^k) = 2T(2^{k-1}) + 2^k$

$T(2^{k-1}) = 2T(2^{k-2}) + 2^{k-1} \quad |\cdot 2$

$T(2^{k-2}) = 2T(2^{k-3}) + 2^{k-2} \quad |\cdot 2^2$

$\vdots$

$T(1) = 1 \quad |\cdot 2^k$

$T(2^k) = \underbrace{2^k + 2^k + 2^k + \dots + 2^k}_{k}$

$T(2^k) = k \cdot 2^k \Rightarrow T(n) = n \cdot \log_2 n \Rightarrow$

$\Theta(n \log_2 n)$

B.

$d'(c) = c \bmod m \quad ; m = 11$

$c_1 = 1, c_2 = 3$

$d(c,i) = (d'(c) + ic_1 + i^2 c_2) \bmod m \Rightarrow$

$\Rightarrow d(c,i) = (c \bmod 11 + i + i^2 \cdot 3) \bmod 11 \rightarrow$ functia de dispersie

c:	0	1	2	3	4	5	6	7	8	9	10
c:	22	-1	-1	17	4	88	28	59	15	31	10

9) pt. 59: $4^0, 8^1, 7^2$

1) pt. 10 : 10

2) pt. 22 : 0

3) pt. 31 : 9

4) pt. 4 : 4

5) pt. 15: $4^0, 8^1$

6) pt. 28 : 6

7) pt. 17 : $6^0, 10^1, 9^2, 3^3$

8) pt. 88 : $0^0, 4^1, 3^2, 8^3, 8^4, 9^5, 3^6$

5

C1. a) $\Theta(2^n)$

$$f(n) = \sum_{i=1}^{n} 2^i = 2 + 2^2 + \dots + 2^n = 2 \cdot \frac{2^n - 1}{2 - 1} = 2^{n+1} - 2 \in \Theta(2^n)$$

C2. e)

În cazul adresării deschise, în momentul ștergerii nu putem marca poziția ca fiind egală (-1), putând astfel risca neidentificarea anumitor elemente din tabelă. Astfel, o opțiune simplă este marcarea cheii cu un argument „ȘTERS" care să indice existența, la un moment dat a unui element pe această poziție.
Operațiile care ține cont sunt:
a) căutarea -> se va continua căutarea dacă dăm peste un element șters, dar se va opri dacă dăm de o poziție egală (-1)
(adăugarea nu ține cont de diferență, dar trebuie actualizată a.î. să poată adăuga și pe poz. șterse)

D.

Arcore

e : TElement[]

dim : Întreg (dimensiunea arcore-ului)

Subalgoritm inaltimi (a, e) este

} pre: a ∈ Arbore
e ∈ TElement, e ≠ NULL_TElement
post: se returnează înălțimea lui e dacă acesta
se află în arbore sau -1 în caz contrar }

Pentru i ← 1, a.dim execută
} se caută elementul în arbore }
Dacă a.e[i] = e atunci
} se determină înălțimea }
i ← -1
creează (c) } creează coadă pt. parcurgerea
arborelui }
adaugă (c, {i, 0})
CâtTimp ¬vidă(c) execută
sterge(c, {poz, nivel})
} se accesează + se elimină elem. din
listă -> pereche (poziție, nivel) }
Dacă nivel > i atunci
i ← nivel
SfDacă

} se continuă BFS }
Dacă poz*2 ≤ a.dim ∧ a.e[poz*2] ≠
NULL_TElement at.
adaugă (c, {poz*2, nivel+1})
SfDacă

Dacă poz*2+1 ≤ a.dim ∧ a.e[poz*2+1] ≠ NULL_TElement
at.
adaugă (c, {poz*2+1, nivel+1})
SfDacă

sf cât timp

{ am det. înălțimea maximului => o returnăm }

înălțime ← i

sf ba că

sf pentru

{ dacă am ajuns aici => elem. nu se află în arbore }

înălțime ← -1

sf subalgoritm

Complexitate:

- de timp : $O(n)$

caz fav - când elem. căutat este în rădăcină => $\Theta(1)$

caz defav - când elem. nu este în arbore => $\Theta(n)$

caz mediu - $\frac{1+2+\dots+n}{2} = \frac{(n+1)n}{2n} \in \Theta(n)$

- de spațiu adițional : $O(n)$