Respostes per a la modalitat **1**

1. Siga la relació de recurrència següent:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T(\frac{n}{2}) + g(n) & \text{altrament} \end{cases}$$

   Si $T(n) \in O(n)$, en quin d'aquests tres casos ens podem trobar?

   (a) $g(n) = \log n$

   [(b)] $g(n) = 1$

   (c) $g(n) = n$

2. Quin d'aquests problemes té una solució eficient utilitzant *programació dinàmica*?

   (a) El problema de l'assignació de tasques.

   (b) La motxilla discreta sense restriccions addicionals.

   [(c)] El problema del canvi (retornar una quantitat de diners amb el mínim nombre de monedes).

3. En els algorismes de *backtracking*, pot el valor d'una fita pessimista ser més gran que el valor d'una fita optimista? (s'entén que ambdues fites s'apliquen sobre el mateix node)

   (a) No, el valor de la fita pessimista d'un node mai pot ser superior al de la fita optimista d'aquest mateix node.

   (b) En general sí, si es tracta d'un problema de maximització, encara que en ocasions els dos valors poden coincidir.

   [(c)] En general sí, si es tracta d'un problema de minimització, encara que en ocasions els dos valors poden coincidir.

4. Per a què pot servir la fita pessimista d'un node de *ramificació i poda*?

   [(a)] Per actualitzar el valor de la millor solució fins al moment.

   (b) Per descartar el node si no és prometedor.

   (c) Per obtenir una fita optimista més precisa.

5. De les expressions següents, o bé dues són certes i una falsa, o bé dues són falses i una certa. Marca la que (en aquest sentit) és diferent a les altres dues.

   (a) $\Theta(\log^2(n)) = \Theta(\log^3(n))$

   (b) $\log(n^3) \notin \Theta(\log_3(n))$

   [(c)] $\Theta(\log_2(n)) = \Theta(\log_3(n))$

6. Donada la relació de recurrència:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ pT(\frac{n}{a}) + g(n) & \text{altrament} \end{cases}$$

(on p i a són enters majors que 1 i $g(n) = n^k$), què ha d'ocórrer perquè es complisca $T(n) \in \Theta(n^k)$?

[(a)] $p < a^k$

(b) $p > a^k$

(c) $p = a^k$

7. La relació de recurrència següent expressa la complexitat d'un algorisme recursiu, on g(n) és una funció polinòmica:

$$T(n) = \begin{cases} 1 & \text{si } n \leq 1 \\ 2T(\frac{n}{2}) + g(n) & \text{altrament} \end{cases}$$

Digues quina de les següents afirmacions és certa:

(a) Si $g(n) \in \Theta(n)$ la relació de recurrència representa la complexitat temporal en el cas pitllor de l'algorisme de de ordenació *quicksort*.

[(b)] Si $g(n) \in \Theta(n)$ la relació de recurrència representa la complexitat temporal en el cas millor de l'algorisme de de ordenació *quicksort*.

(c) Si $g(n) \in \Theta(1)$ la relació de recurrència representa la complexitat temporal en el cas millor de l'algorisme d'ordenació *mergesort*.

8. Quina és la complexitat temporal de la següent funció?

```
unsigned exam (unsigned n) {
unsigned i=n, k=0;
   while (i>0){
       unsigned j=i;
       do{
          j = j * 2;
          k = k + 1;
       } while (j<=n);
       i = i / 2;
   }
   return k;
}
```

(a) $\Theta(\log n)$

[(b)] $\Theta(\log^2 n)$

(c) $\Theta(n)$

9. Quina és la complexitat temporal en el millor dels casos de la següent funció?

```
void exam (vector <int >& v){
int i=0,j,x,n=v.size();
bool permuta=1;
    while (n>0 && permuta){
       i=i+1;
       permuta=0;
       for (j=n-1;j>=i;j--)
          if (v[j] <v[j-1]){
             x=v[j];
             permuta=1;
             v[j]=v[j-1];
             v[j-1]=x;
          }
    }
}
```

(a) $\Omega(1)$

(b) $\Omega(n)$

(c) Aquesta funció no té cas millor.

10. Davant un problema d'optimització resolt mitjançant *backtracking*, pot ocórrer que l'ús de les fites pessimistes i optimistes siga inútil, o fins i tot perjudicial?

(a) Sí, ja que és possible que a pesar d'utilitzar aquestes fites no es descarte cap node.

(b) Segons el tipus de fita; les pessimistes pot ser que no descarten cap node però l'ús de fites optimistes garanteix la reducció l'espai de recerca.

(c) No, les fites tant optimistes com a pessimistes garanteixen la reducció de l'espai de solucions i per tant l'eficiència de l'algorisme.

11. Indiqueu quina d'aquestes afirmacions és *certa*.

(a) La memoïtzació evita que un algorisme recursiu ingenu resolga repetidament el mateix problema.

(b) L'avantatge de la solució de programació dinàmica iterativa al problema de la motxilla discreta és que mai es realitzen càlculs innecessaris.

(c) Els algorismes iteratius de programació dinàmica utilitzen memoïtzació per evitar resoldre de nou els mateixos subproblemes que es tornen a presentar.

12. Per a quin d'aquests problemes d'optimització es coneix almenys una solució voraç?

    [(a)] L'arbre de recobriment mínim per a un graf no dirigit amb pesos.

    (b) El problema de la motxilla discreta.

    (c) El problema de l'assignació de cost mínim de n tasques a n treballadors quan el cost d'assignar la tasca i al treballador j, c_{ij} està tabulat en una matriu.

13. Garanteix l'ús d'una estratègia "divideix i venceràs" l'existència d'una solució de complexitat temporal polinòmica a qualsevol problema?

    [(a)] No

    (b) Sí, en qualsevol cas.

    (c) Sí, però sempre que la complexitat temporal conjunta de les operacions de descomposició del problema i la combinació de les solucions siga polinòmica.

14. La complexitat temporal de la solució via tornada arrere al problema de la motxilla discreta sense fraccionament és...

    [(a)] ... exponencial en el cas pitjor.

    (b) ... quadràtica en el cas pitjor.

    (c) ... exponencial en qualsevol cas.

15. Quin dels criteris següents proporcionaria una fita optimista per al problema de trobar el camí mes curt entre dues ciutats (se suposa que el graf és connex)?.

    (a) Calcular la distància recorreguda movent-se a l'atzar pel graf fins a arribar (per atzar) a la ciutat de destinació.

    [(b)] Calcular la distància geomètrica (en línia recta) entre les ciutats d'origen i de destinació.

    (c) Utilitzar la solució (subóptima) que s'obté quan es resol el problema mitjançant un algorisme voraç.

16. Es vol reduir la complexitat temporal de la següent funció fent ús de programació dinàmica. Quin seria la complexitat temporal resultant?

```
unsigned g( unsigned n, unsigned r){
   if (r==0 || r==n)
      return 1;
   return g(n-1, r-1) + g(n-1, r);
}
```

    (a) Es pot reduir fins a lineal.

    [(b)] Cuadrática

    (c) La funció no compleix amb els requisits necessaris per a poder aplicar programació dinàmica.

17. Si $\lim_{n\to\infty}(f(n)/n^2) = k$, i $k \neq 0$, quina d'aquestes tres afirmacions és *certa*?

    [(a)] $f(n) \in \Theta(n^2)$

    (b) $f(n) \in \Omega(n^3)$

    (c) $f(n) \in \Theta(n^3)$

18. Siga A una matriu quadrada $n \times n$. Es tracta de buscar una permutació de les columnes tal que la suma dels elements de la diagonal de la matriu resultant siga mínima. Indiqueu quina de les següents afirmacions és certa.

    (a) La complexitat temporal de la millor solució possible al problema és $O(n \log n)$.

    (b) La complexitat temporal de la millor solució possible al problema està en $\Omega(n^n)$.

    [(c)] Si es construeix una solució al problema basada en l'esquema de ramificació i poda, una bona elecció de fites optimistes i pessimistes podria evitar l'exploració de totes les permutacions possibles.

19. De les expressions següents, o bé dues són certes i una falsa, o bé dues són falses i una certa. Marca la que (en aquest sentit) és diferent a les altres dues.

    [(a)] $\Theta(f) = O(f) \cap \Omega(f)$

    (b) $O(f) = \Omega(f) \cap \Theta(f)$

    (c) $\Omega(f) = \Theta(f) \cap O(f)$

20. De les expressions següents, o bé dues són certes i una falsa, o bé dues són falses i una certa. Marca la que (en aquest sentit) és diferent a les altres dues.

    [(a)] $O(2^{\log_2(n)}) \subset O(n^2) \subset O(n!)$

    (b) $O(n^2) \subset O(2^{\log_2(n)}) \subset O(2^n)$

    (c) $(4^{\log_2(n)}) \subset O(n) \subset O(2^n)$

21. Quan es resol el problema de la motxilla discreta usant l'estratègia de tornada arrere, pot ocórrer que es tarde menys a trobar la solució òptima si es prova primer a ficar cada objecte abans de no ficar-ho?

    [(a)] Sí, però només si s'usen fites optimistes per podar l'arbre de recerca.

    (b) Sí, tant si s'usen fites optimistes per podar l'arbre de recerca com si no.

    (c) No, ja que en qualsevol cas s'han d'explorar totes les solucions factibles.

22. En els algorismes de *ramificació i poda* ...

    [(a)] Una fita optimista és necessàriament un valor insuperable; si no fóra així es podria podar el node que condueix a la solució òptima.

    (b) Una fita optimista és necessàriament un valor assolible; si no és així no està garantit que es trobe la solució òptima.

    (c) Una fita pessimista és el valor que com a màxim aconsegueix qualsevol node factible que no és l'òptim.

23. En un algorisme de *ramificació i poda*, l'ordre escollit per prioritzar els nodes en la llista de nodes vius ...

    (a) ... mai afecta al temps necessari per trobar la solució òptima.

    (b) ... determina la complexitat temporal en el pitjor dels casos de l'algorisme.

    [(c)] ... pot influir en el nombre de nodes que es descarten sense arribar a expandir-los.

24. Els algorismes de *tornada arrere* que fan ús de fites optimistes generen les solucions possibles al problema mitjançant ...

    (a) ... un recorregut guiat per les que poden ser les millors branques de l'arbre que representa l'espai de solucions.

    [(b)] ... un recorregut en profunditat de l'arbre que representa l'espai de solucions.

    (c) ... un recorregut guiat per una cua de prioritat d'on s'extrauen primer els nodes que representen els subarbres més prometedors de l'espai de solucions.

25. Què s'entén per *grandària del problema*?

    [(a)] La quantitat d'espai en memòria que es necessita per codificar una instància d'aquest problema.

    (b) El valor màxim que pot prendre una instància qualsevol d'aquest problema.

    (c) El nombre de paràmetres que componen el problema.

26. L'algorisme d'ordenació *Quicksort* divideix el problema en dos subproblemes. Quina és la complexitat temporal asimptòtica de realitzar aquesta divisió?

    (a) $O(1)$

    [(b)] $O(n)$

    (c) $O(n \log n)$

27. Es vol ordenar d números diferents compresos entre 1 i n. Per a això s'usa un array de n booleans que s'inicialitzen primer a *false*. A continuació es recorren els d números canviant els valors de l'element del vector de booleans corresponent al seu números a *true*. Finalment es recorre el vector de booleans escrivint els índexs dels elements del vector de booleans que són *true*. És aquest algorisme més ràpid (asimptòticament) que el *mergesort*?

   [(a)] Només si $d \log d > k\,n$ (on k és una constant que depén de la implementació)

   (b) Sí, ja que el *mergesort* és $O(n \log n)$ i aquest és $O(n)$

   (c) No, ja que aquest algorisme ha de recórrer diverses vegades el vector de booleans.

28. En el problema de l'acolorit de grafs (mínim nombre de colors necessaris per a acolorir tots els vèrtexs d'un graf de manera que no queden dos adjacents amb el mateix color) resolt mitjançant *ramificació i poda*, de quina manera s'hauria d'ordenar la llista de nodes vius per a obtenir una solució acceptable?

   (a) Per cota optimista: explorant primer els nodes amb menor cota optimista.

   (b) Per cota pessimista: explorant primer els nodes amb menor cota optimista.

   [(c)] les altres dues opcions poden ser ambdues correctes.

29. Quina és la definició correcta de $\Omega(g)$?

   [(a)] $\Omega(g) = \{f : \mathbb{N} \to \mathbb{R}^+ | \exists c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, f(n) \geq cg(n)\}$

   (b) $\Omega(g) = \{f : \mathbb{N} \to \mathbb{R}^+ | \exists c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \geq cf(n)\}$

   (c) $\Omega(g) = \{f : \mathbb{N} \to \mathbb{R}^+ | \forall c \in \mathbb{R}, \exists n_0 \in \mathbb{N}, \forall n \geq n_0, g(n) \geq cf(n)\}$

30. La solució recursiva ingènua (però correcta) a un problema d'optimització crida més d'una vegada a la funció amb els mateixos paràmetres. Una de les tres afirmacions següents és falsa.

   [(a)] Es pot millorar l'eficiència de l'algorisme convertint l'algorisme recursiu directament en iteratiu sense canviar el seu funcionament bàsic.

   (b) Es pot millorar l'eficiència de l'algorisme guardant en una taula el valor retornat per a cada conjunt de paràmetres de cada crida quan aquesta es produeix per primera vegada.

   (c) Es pot millorar l'eficiència de l'algorisme definint per endavant l'ordre en el qual s'han de calcular les solucions als subproblemes i omplint una taula en aquest ordre.

31. En l'esquema de *tornada enrere,* els mecanismes de poda basats en la millor solució fins al moment...

    [(a)] ... poden eliminar solucions parcials que són factibles.

    (b) ... garanteixen que no s'explorarà mai tot l'espai de solucions possibles.

    (c) Les altres dues opcions són ambdues certes.

32. En el problema de l'acolorit de grafs (mínim nombre de colors necessaris per a acolorir tots els vèrtexs d'un graf de manera que no queden dos adjacents amb el mateix color) resolt mitjançant *ramificació i poda,* una cota optimista és el resultat de l'assumir que ...

    [(a)] ...no es van a utilitzar colors diferents als ja utilitzats.

    (b) ...es van a utilitzar tants colors diferents als ja utilitzats com a vèrtexs queden per acolorir.

    (c) ...només vaa ser necessari un color més.

33. Considerem l'algorisme d'ordenació *Mergesort* modificat de manera que, en comptes de dividir el vector en dues parts, es divideix en tres. Posteriorment es combinen les solucions parcials. Quina seria la complexitat temporal asimptòtica de la combinació de les solucions parcials?

    (a) Cap de les altres dues opcions és certa.

    [(b)] $\Theta(n)$

    (c) $\Theta(\log_3 n)$

34. Es vol reduir la complexitat temporal de la funció `g` usant programació dinàmica iterativa. Quina seria la complexitat espacial de l'algorisme resultant?

```
int g( int p[], unsigned n ) {
   if (n==0)
      return 0;
   int q = -1;
   for ( unsigned i = 1; i <= n; i++ )
      q = max( q, p[i] + g( p, n-i ) );
   return q;
}
```

    (a) Quadràtica

    [(b)] Lineal

    (c) Cúbica

35. Si $f \in \Theta(g_1)$ y $f \in \Theta(g_2)$ aleshores

    (a) $f \notin \Theta(\text{máx}(g_1, g_2))$

    (b) $f \in \Theta(g_1 \cdot g_2)$

    [(c)] $f \in \Theta(g_1 + g_2)$