Respostes per a la modalitat **2**

1. 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) $\Theta(\log(n^2)) = \Theta(\log(n^3))$

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

2. Donades les funcions següents:

```
// Precondició: { 0 <= i < v.size(); i < j <= v.size() }
unsigned f( const vector<unsigned>&v, unsigned i, unsigned j ) {
  if( i == j+1 )
    return v[i];
  unsigned sum = 0;
  for( unsigned k = 0; k < j - i; k++ )
    sum += f( v, i, i+k+1) + f( v, i+k+1, j );
  return sum;
}

unsigned g( const vector<unsigned>&v ) {
  return f( v, v.begin(), v.end() );
}
```

   Es vol reduir la complexitat temporal de la funció `g` usant programació dinàmica iterativa. quina seria la complexitat espacial?

   [(a)] quadràtica

   (b) cúbica

   (c) Es pot reduir fins a lineal

3. 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.

4. 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.

5. Quan s'usa un algorisme voraç per abordar la resolució d'un problema d'optimització per selecció discreta (és a dir, un problema pel qual la solució consisteix a trobar un subconjunt del conjunt d'elements que optimitza una determinada funció), quina d'aquestes tres coses és impossible que ocórrega?

   [(a)] Que es reconsidere la decisió ja presa anteriorment quant a la selecció d'un element a la vista de la decisió que s'ha de prendre en un instant.

   (b) Que l'algorisme no trobe cap solució.

   (c) Que la solució no siga l'òptima.

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 $T(n) \in \Theta(n^k \log_a(n))$?

   (a) $p < a^k$

   (b) $p > a^k$

   [(c)] $p = a^k$:

7. 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.

8. Els algorismes voraços...

   [(a)] ... es basen en el fet que la solució óptima es pot construir afegint repetidament el millor element dels que queden.

   (b) ... es basen en el fet que les dades estan organitzades de tal manera que la meitat de les dades es poden descartar en cada pas.

   (c) es basen en el fet que es pot estalviar esforç guardant els resultats de càlculs anteriors en una taula.

9. 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.

10. En l'esquema de *tornada arrere* l'ordre en el qual es van assignant els diferents valors a les components del vector que contindrà la solució...

    (a) ... és irrellevant si no s'utilitzen mecanismes de poda basats en la millor solució fins al moment.

    (b) ... pot ser rellevant si s'utilitzen mecanismes de poda basats en estimacions optimistes.

    [(c)] Les dues anteriors són certes. se

11. 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 falsa.

    [(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 $\Omega(n^2)$.

    (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.

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. 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).

14. 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.

15. Quina és la definició correcta de $\Omega(f)$?

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

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

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

16. 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 dos anteriors són veritables.

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

    (a) $f \in \Theta(\max(g_1, g_2))$

    (b) $f^2 \in \Theta(g_1 \cdot g_2)$

    [(c)] Les altres dues opcions són ambdues certes.

18. 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.

19. 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.

20. 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.

21. 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 $T(n) \in O(n^2)$, en quin d'aquests tres casos ens podem trobar?

    (a) $g(n) = n$
    [(b)] $g(n) = n^2$
    (c) $g(n) = n \log n$

22. L'ús de funcions de fita en ramificació i poda ...

    (a) ... transforma en polinòmiques complexitats que abans eren exponencials.
    [(b)] ... pot reduir el nombre d'instàncies del problema que pertanyen al cas pitjor.
    (c) ... garanteix que l'algorisme serà més eficient davant qualsevol instància del problema.

23. El programa següent resol el problema de tallar un tub de longitud n en segments de longitud sencera entre 1 i n de manera que es maximitze el preu d'acord amb una taula que dóna el preu per a cada longitud, però falta un tros. Què hauria d'anar-hi en comptes de **XXXXXXX**?

```
void fill(price m[]) {
for (index i=0;i<=n;i++) m[i]=-1;
}

price cutrod(length n, price m[], price p[]) {
price q;
if (m[n]>=0) return m[n];
if (n==0) q=0;
else {
       q=-1;
       for (index i=1;i<=n;i++)
           q=max(q,p[i]+cutrod(XXXXXXX));
}
m[n]=q;
return q;
}
```

    [(a)] `n-i,m,p`
    (b) `n,m[n]-1,p`
    (c) `n-m[n],m,p`

24. 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.

25. 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.

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

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

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

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

27. 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) $O(2^{\log_2(n)}) \subseteq O(n^2) \subset O(n!)$

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

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

28. Per a un d'aquests tres problemes no es coneix una solució trivial i eficient que seguisca l'esquema voraç.

    (a) El problema de la motxilla discreta sense limitació en la càrrega màxima de la motxilla.

    (b) El problema de la motxilla contínua.

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

29. Indiqueu quina d'aquestes afirmacions és falsa.

    (a) Hi ha problemes d'optimització per als quals es pot obtenir sempre la solució òptima utilitzant una estratègia voraç.

    (b) Hi ha problemes d'optimització en els quals el mètode voraç només obté la solució òptima per a algunes instàncies i un subòptim per a moltes altres instàncies.

    [(c)] Tots els problemes d'optimització tenen una solució voraç que és òptima sigui el que sigui la instància a resoldre.

30. 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.

31. Quan la descomposició recursiva d'un problema dóna lloc a subproblemes de grandària similar a l'original, quin esquema promet ser més apropiat?en

    (a) Divideix i venceràs, sempre que es garantisca que els subproblemes no són de la mateixa grandària.

    [(b)] Programació dinàmica.

    (c) El mètode voraç

32. 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 del nou algorisme?

    (a) $n^2 \log(n)$

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

    (c) $n \log^2(n)$

33. 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.

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

    [(a)] $O(1)$

    (b) $O(n)$

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

35. 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.

36. Si un problema d'optimització ho és per a una funció que pren vspalors continus ...

    [(a)] ...la programació dinàmica recursiva pot resultar molt més eficient que la programació dinàmica iterativa quant a l'ús de memòria.

    (b) ...la programació dinàmica iterativa sempre és molt més eficient que la programació dinàmica recursiva quant a l'ús de memòria.

    (c) ...l'ús de memòria de la programació dinàmica iterativa i de la programació dinàmica recursiva és el mateix independentment de si el domini és discret o continu.

37. La funció γ d'un nombre semienter positiu (un nombre és semienter si en restar-li 0,5 és sencer) es defineix com:
parcial

```
double gamma( double n ) { // S'hi assumeix n>=0.5 y n-0.5
enter
  if( n == 0.5 )
    return sqrt(PI);
  return n * gamma( n - 1);
}
```

Es pot calcular usant programació dinàmica iterativa? co

(a) Sí, però la complexitat temporal no millora.

(b) No, ja que l'índex del magatzem seria un nombre real i no enter.

(c) No, ja que no podríem emmagatzemar els resultats intermedis en el magatzem.

38. Indiqueu quina d'aquestes afirmacions és falsa.

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

(b) La solució de programació dinàmica iterativa al problema de la motxilla discreta realitza càlculs innecessaris.

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

39. Si el següent és un esquema general per resoldre problemes de maximització, què falta en el buit?:

```
Solution BB( Problem p ) {
Node best, init = initialNode(p);
Value pb = init.pessimistic_b();
priority_queue<Node>q;
 q.push(init);
 while( ! q.empty() ) {
    Node n = q.top(); q.pop();
    q.pop();
    if( ????????? ){
     pb = max( pb, n.pessimistic_b());
     if( n.isTerminal() )
        best = n.sol();
     else
        for( Node n : n.expand() )
           if( n.isFeasible())
              q.push(n);
 }
return best;
}
```

(a) `n.optimistic_b() <= pb`

(b) `n.pessimistic_b() <= pb`

(c) `n.optimistic_b() >= pb`

40. 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 falsa:

(a) Si $g(n) \in \Theta(n)$ la relació de recurrència representa la complexitat temporal de l'algorisme d'ordenació *mergesort*.

(b) Si $g(n) \in \Theta(1)$ la relació de recurrència representa la complexitat temporal de l'algorisme de recerca dicotòmica.

(c) 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*.