EJEMPLOS DE PROBLEMAS RESUELTOS

UNIVERSIDAD DE MÁLAGA
E.T.S. INGENIERÍA INFORMÁTICA

Fco Javier Moreno Barea

1. Resolver el problema del cambio de monedas en el caso en el que el número de monedas de cada tipo es limitado. Supóngase que hay k tipos de monedas con denominaciones d_i, cada uno con n_i monedas disponibles. M es la cantidad que se desea conseguir con el número mínimo de monedas.

   a) Definir la estructura de los subproblemas, el problema inicial, y el proceso de ramificación.

   b) Determinar una forma general de calcular la cota inferior de la solución alcanzable para un subproblema.

   c) Construir a partir de lo anterior el árbol de búsqueda que exploraría el algoritmo B&B con la estrategia best-first para la siguiente instancia: $M = 41$, denominaciones $d_i \in \langle 1, 2, 5, 10, 20 \rangle$, con $n_i \in \langle 0, 5, 5, 2, 2 \rangle$ monedas disponibles de cada tipo.

a) Definir la estructura de los subproblemas, el problema inicial, y el proceso de ramificación.

Contenido de un estado:

- Vector con los k tipos de monedas, o denominaciones (tamaño k), y la cantidad restante a devolver
  - En el vector V[j] = i significa que de la moneda de tipo j se utiliza un número i.
  - La cantidad restante a devolver es un número.

- Estado inicial: las k componentes están a 0 ($\forall j = 0, \dots, k-1 : V[j] = 0$); y la cantidad restante a devolver es la inicial (M).

- Estado intermedio: algunas componentes desde la 0 hasta la k tienen valores distintos de 0 (se ha hecho uso de una moneda de ese tipo). la cantidad restante a devolver es la inicial menos el valor de las monedas utilizadas por el número de monedas de cada tipo utilizada ($M - \left[\sum_{i=0}^{k} V[i] * d_i\right]$)

- Estado final: algunas de las componentes son distintas de 0 y la cantidad restante a devolver es 0.

a) Definir la estructura de los subproblemas, el problema inicial, y el proceso de ramificación.

Proceso de Ramificación:

- Se puede ramificar añadiendo 1 a cada tipo de moneda siempre y cuando el número utilizado no sea mayor que el número máximo de monedas que hay (n_i); y siempre la cantidad restante a devolver actualizada según la moneda a utilizar no sea menor que 0.

- De manera que en el nivel 1, se rellena la componente 0 con los valores 1…k, de izquierda a derecha, siempre que sea posible, con lo que la ramificación será de máximo k.

b) Determinar una forma general de calcular la cota inferior de la solución alcanzable para un subproblema. ¿Cuál es la complejidad del cálculo de la cota?

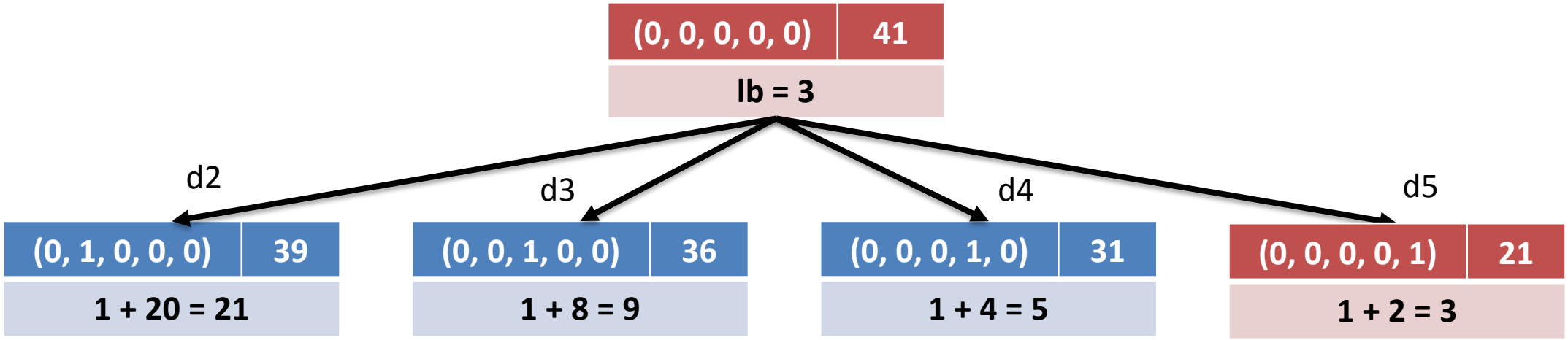
- Cálculo de la cota: f(x) = g(x) + h(x) donde,
  - g(x) = número de monedas ya utilizadas ($\sum_{i=0}^{k-1} V[i]$).
  - h(x) = mínimo número de monedas del tipo i necesario para pagar el resto (R) a devolver ($R/d_i + 1$).

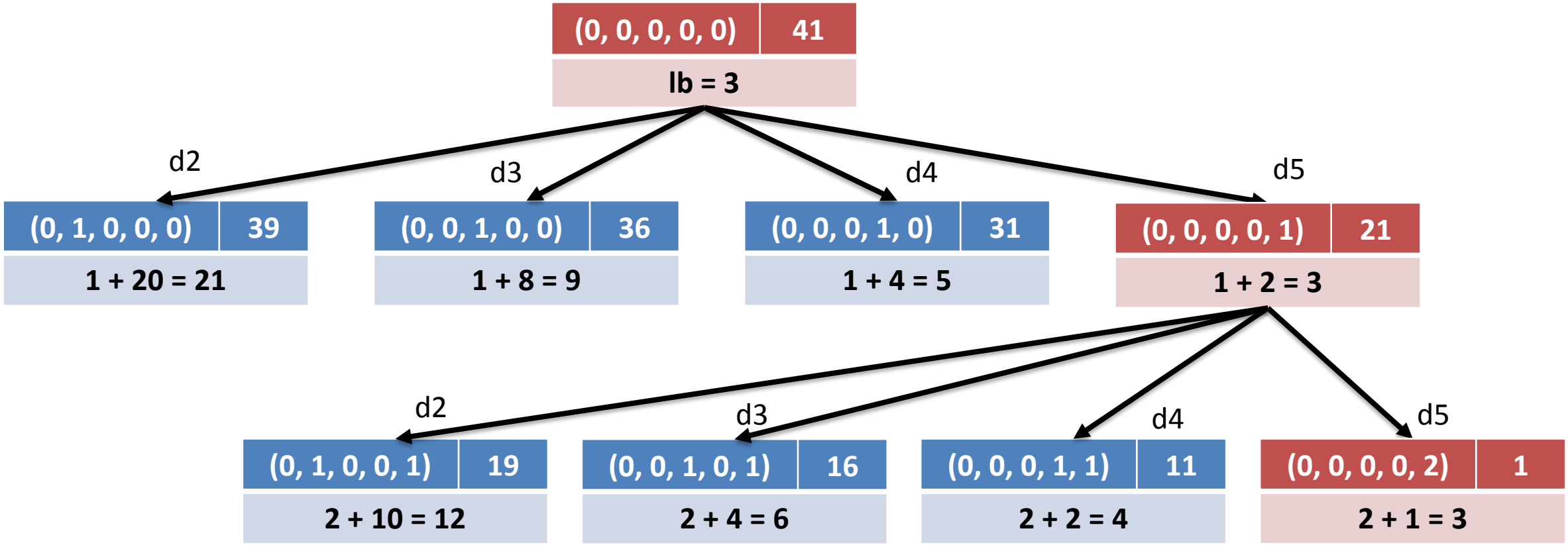
- Por ejemplo, nodo 0 (inicial): número de monedas ya utilizadas es nulo, g(0) = 0; estimación optimista de mínimo número de monedas es h(0) = 41/20 + 1 = 3. Por lo que f(0) = g(0) + h(0) = 3.

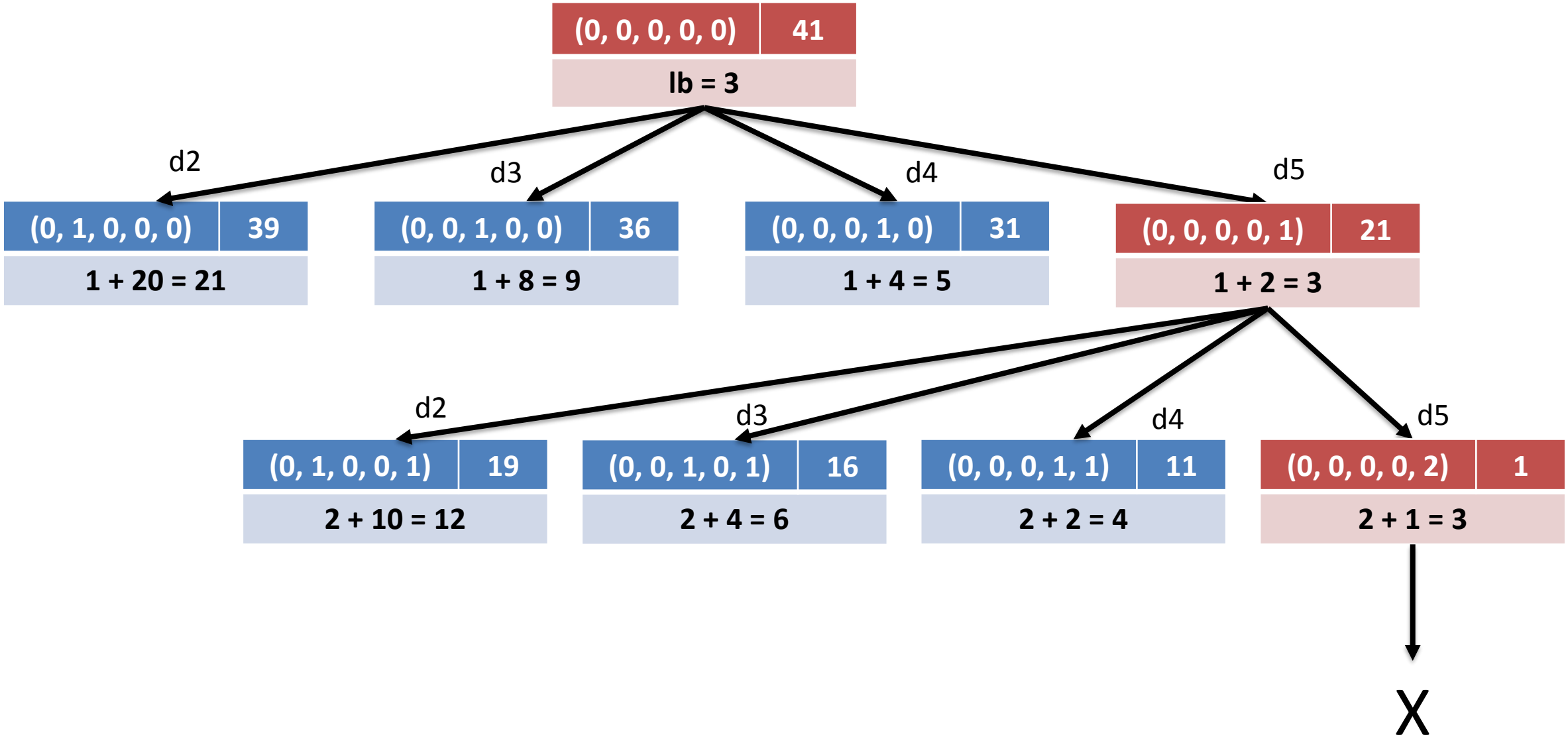
- Complejidad del cálculo de la cota: Lineal.

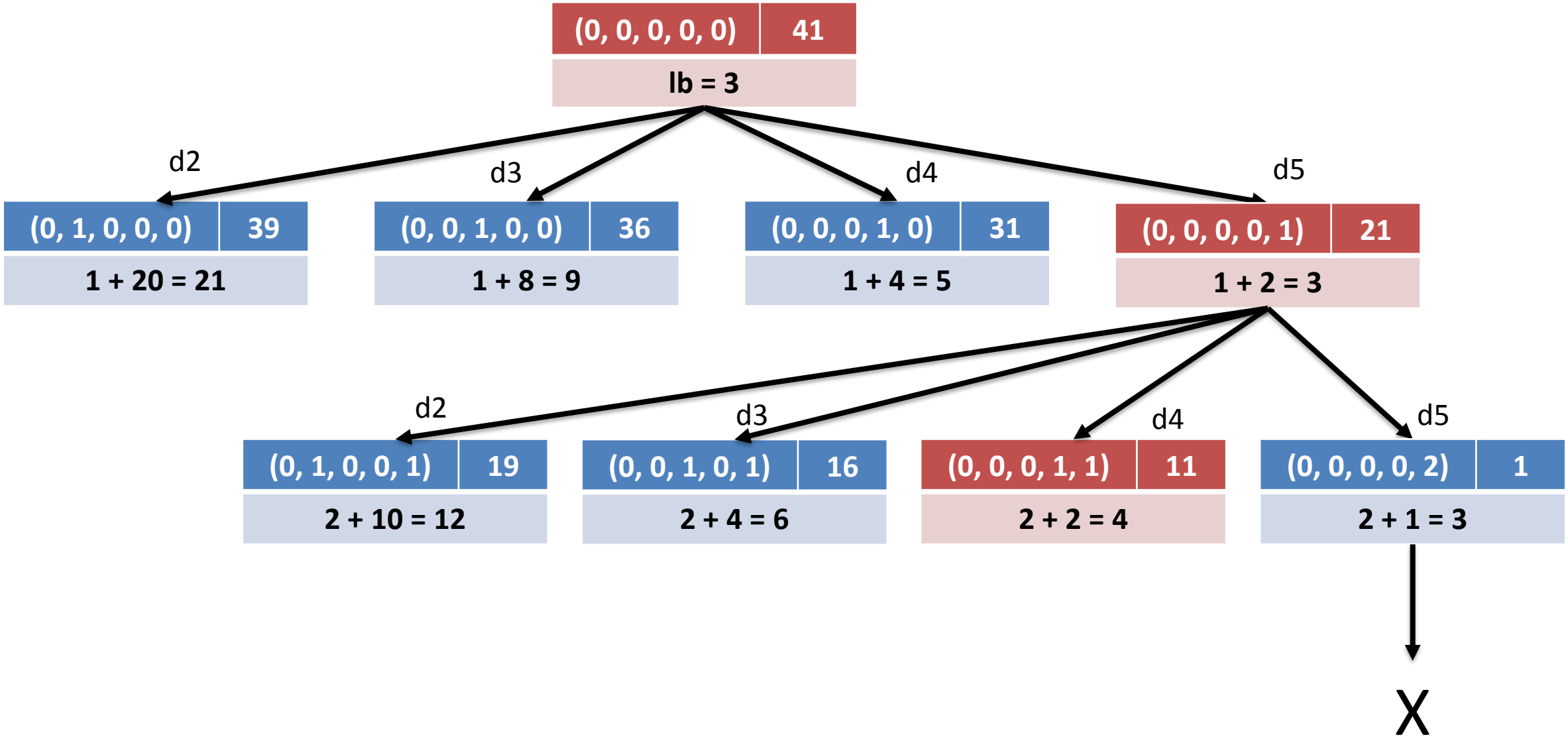
c) Construir a partir de lo anterior el árbol de búsqueda que exploraría el algoritmo B&B con la estrategia best-first para la siguiente instancia: $M = 41$, denominaciones $d_i \in \langle 1, 2, 5, 10, 20 \rangle$, con $n_i \in \langle 0, 5, 5, 2, 2 \rangle$ monedas disponibles de cada tipo.

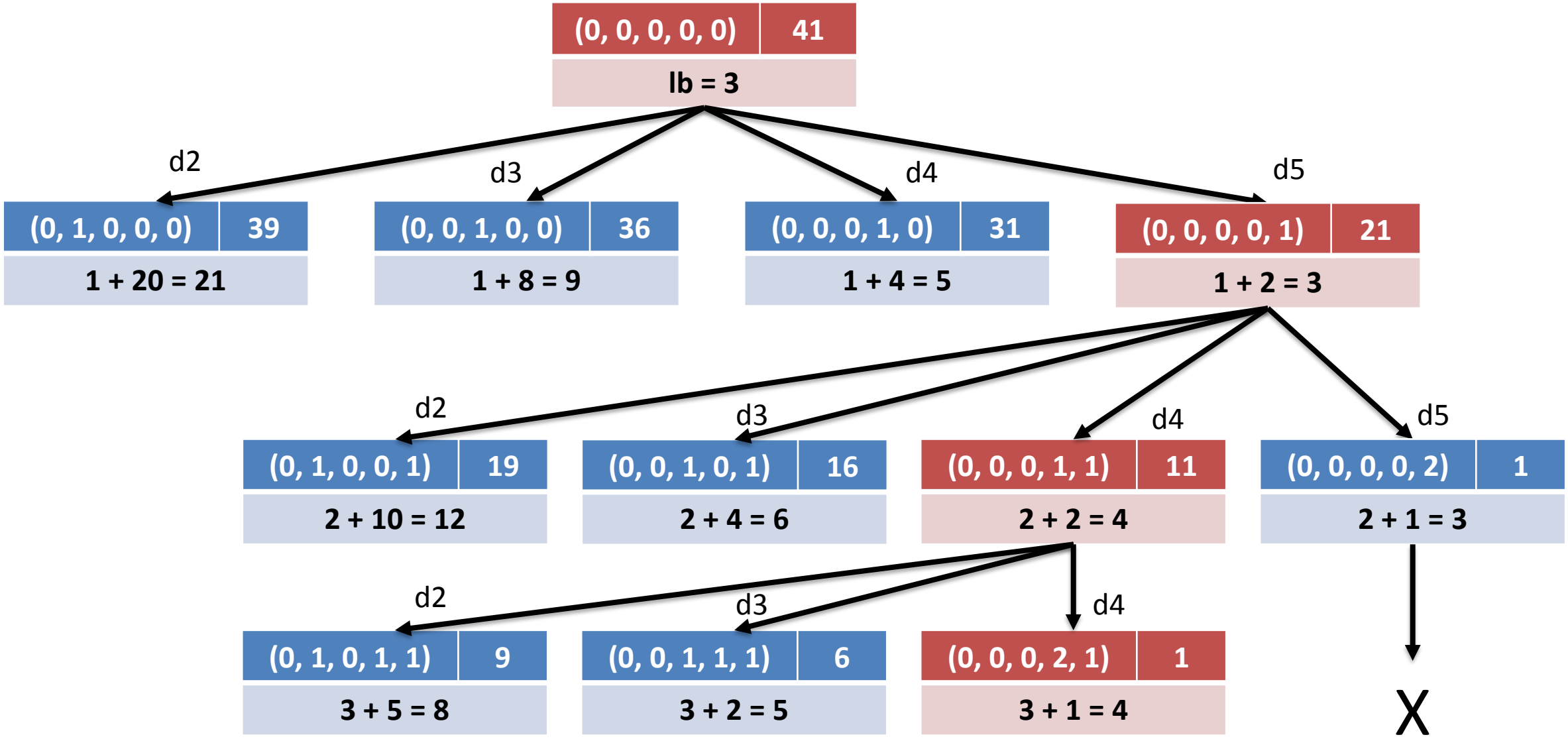
(0, 0, 0, 0, 0)	41
lb = 3	

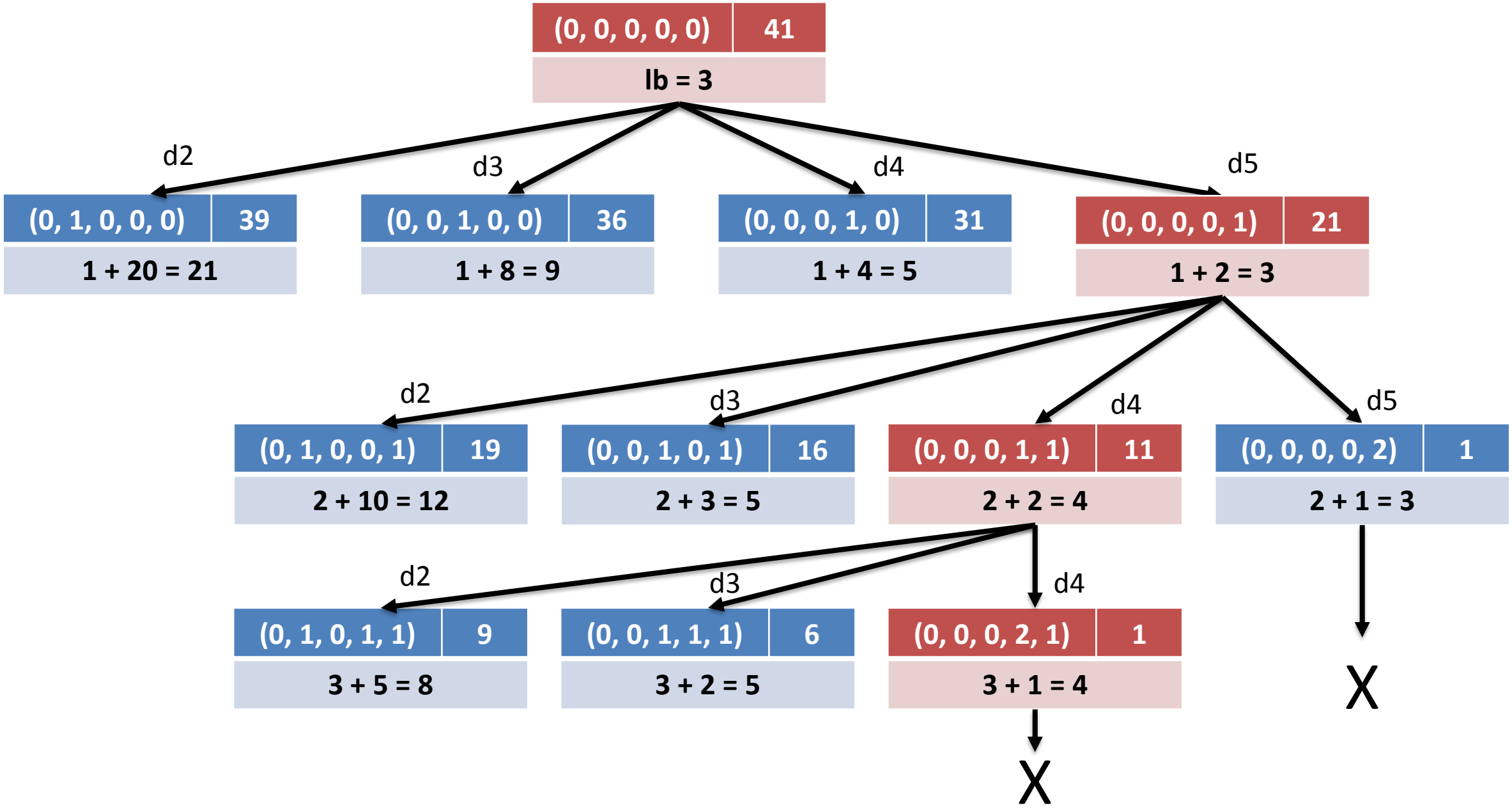

(0, 0, 0, 0, 0)
41
lb = 3
d2
d3
d4
d5
(0, 1, 0, 0, 0)
39
1 + 20 = 21
(0, 0, 1, 0, 0)
36
1 + 8 = 9
(0, 0, 0, 1, 0)
31
1 + 4 = 5
(0, 0, 0, 0, 1)
21
1 + 2 = 3

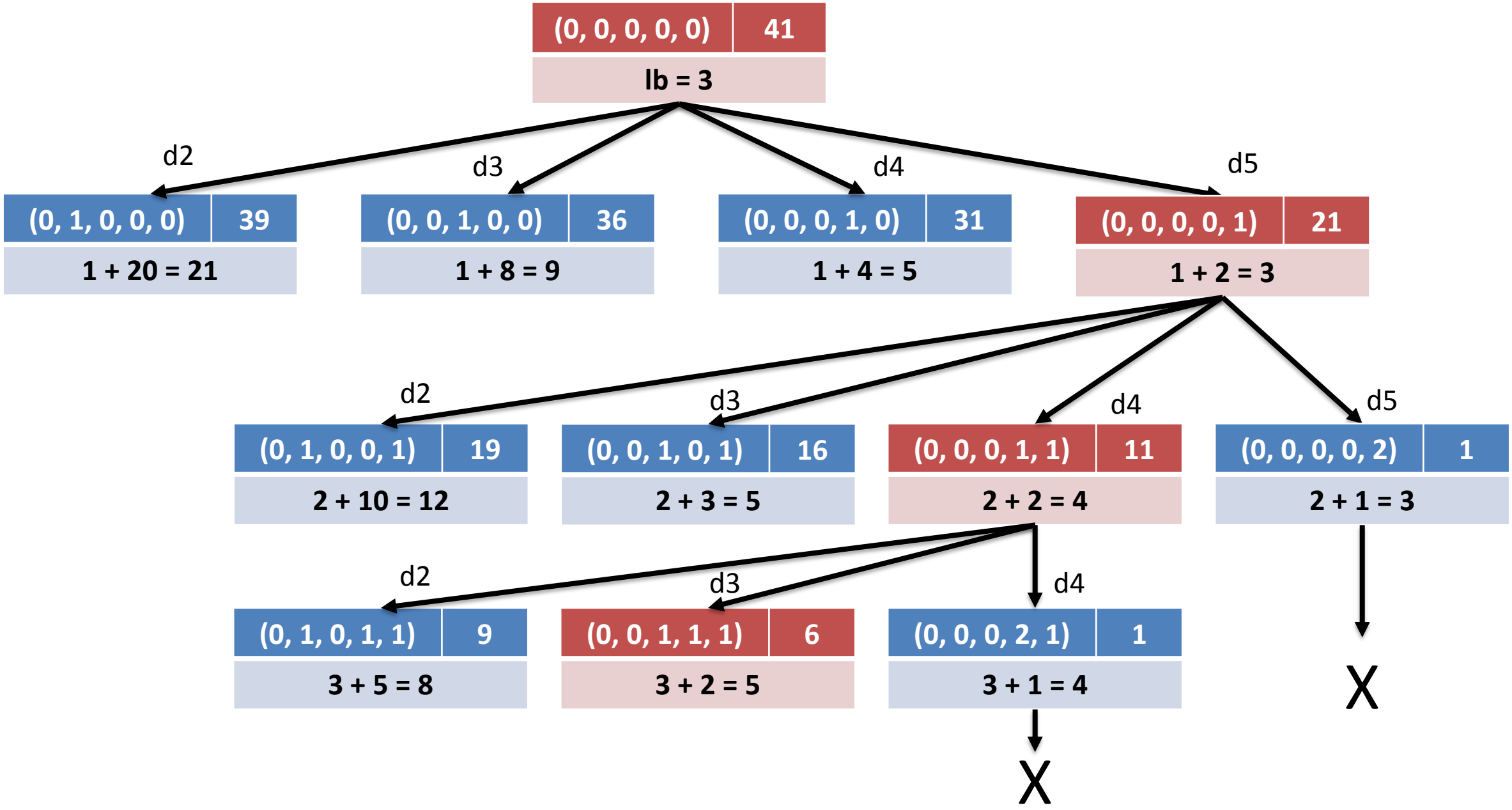

(0, 0, 0, 0, 0)
41
lb = 3
d2
d3
d4
d5
(0, 1, 0, 0, 0)
39
1 + 20 = 21
(0, 0, 1, 0, 0)
36
1 + 8 = 9
(0, 0, 0, 1, 0)
31
1 + 4 = 5
(0, 0, 0, 0, 1)
21
1 + 2 = 3
d2
d3
d4
d5
(0, 1, 0, 0, 1)
19
2 + 10 = 12
(0, 0, 1, 0, 1)
16
2 + 4 = 6
(0, 0, 0, 1, 1)
11
2 + 2 = 4
(0, 0, 0, 0, 2)
1
2 + 1 = 3


(0, 0, 0, 0, 0)
41
lb = 3
d2
d3
d4
d5
(0, 1, 0, 0, 0)
39
1 + 20 = 21
(0, 0, 1, 0, 0)
36
1 + 8 = 9
(0, 0, 0, 1, 0)
31
1 + 4 = 5
(0, 0, 0, 0, 1)
21
1 + 2 = 3
d2
d3
d4
d5
(0, 1, 0, 0, 1)
19
2 + 10 = 12
(0, 0, 1, 0, 1)
16
2 + 4 = 6
(0, 0, 0, 1, 1)
11
2 + 2 = 4
(0, 0, 0, 0, 2)
1
2 + 1 = 3
X


(0, 0, 0, 0, 0)
41
lb = 3
d2
d3
d4
d5
(0, 1, 0, 0, 0)
39
1 + 20 = 21
(0, 0, 1, 0, 0)
36
1 + 8 = 9
(0, 0, 0, 1, 0)
31
1 + 4 = 5
(0, 0, 0, 0, 1)
21
1 + 2 = 3
d2
d3
d4
d5
(0, 1, 0, 0, 1)
19
2 + 10 = 12
(0, 0, 1, 0, 1)
16
2 + 4 = 6
(0, 0, 0, 1, 1)
11
2 + 2 = 4
(0, 0, 0, 0, 2)
1
2 + 1 = 3
X


(0, 0, 0, 0, 0)
41
lb = 3
d2
d3
d4
d5
(0, 1, 0, 0, 0)
39
1 + 20 = 21
(0, 0, 1, 0, 0)
36
1 + 8 = 9
(0, 0, 0, 1, 0)
31
1 + 4 = 5
(0, 0, 0, 0, 1)
21
1 + 2 = 3
d2
d3
d4
d5
(0, 1, 0, 0, 1)
19
2 + 10 = 12
(0, 0, 1, 0, 1)
16
2 + 4 = 6
(0, 0, 0, 1, 1)
11
2 + 2 = 4
(0, 0, 0, 0, 2)
1
2 + 1 = 3
d2
d3
d4
(0, 1, 0, 1, 1)
9
3 + 5 = 8
(0, 0, 1, 1, 1)
6
3 + 2 = 5
(0, 0, 0, 2, 1)
1
3 + 1 = 4
X


(0, 0, 0, 0, 0)
41
lb = 3
d2
d3
d4
d5
(0, 1, 0, 0, 0)
39
1 + 20 = 21
(0, 0, 1, 0, 0)
36
1 + 8 = 9
(0, 0, 0, 1, 0)
31
1 + 4 = 5
(0, 0, 0, 0, 1)
21
1 + 2 = 3
d2
d3
d4
d5
(0, 1, 0, 0, 1)
19
2 + 10 = 12
(0, 0, 1, 0, 1)
16
2 + 3 = 5
(0, 0, 0, 1, 1)
11
2 + 2 = 4
(0, 0, 0, 0, 2)
1
2 + 1 = 3
d2
d3
d4
(0, 1, 0, 1, 1)
9
3 + 5 = 8
(0, 0, 1, 1, 1)
6
3 + 2 = 5
(0, 0, 0, 2, 1)
1
3 + 1 = 4
X
X

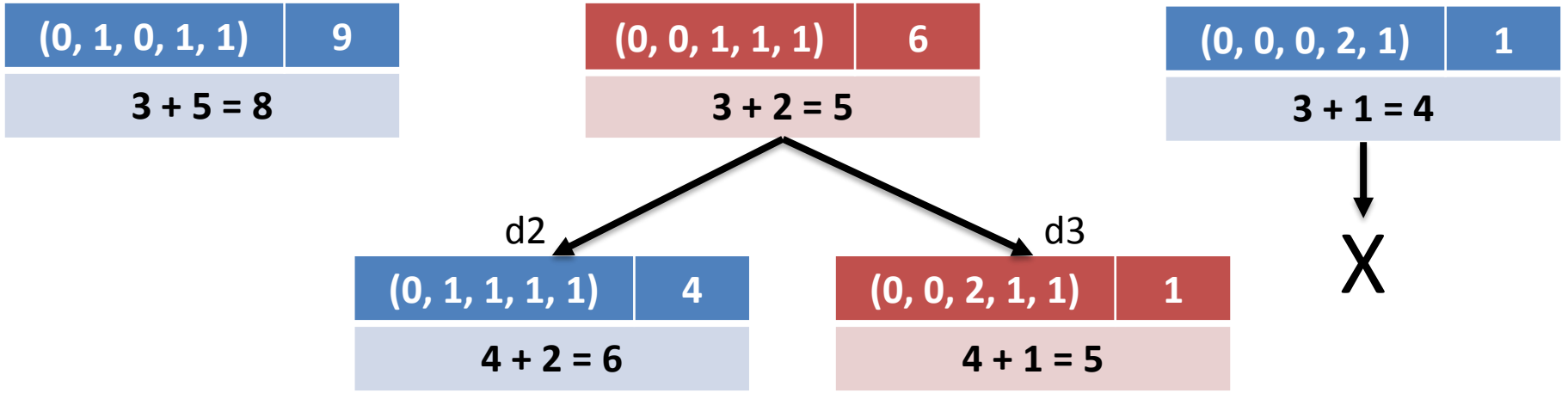

(0, 0, 0, 0, 0)
41
lb = 3
d2
d3
d4
d5
(0, 1, 0, 0, 0)
39
1 + 20 = 21
(0, 0, 1, 0, 0)
36
1 + 8 = 9
(0, 0, 0, 1, 0)
31
1 + 4 = 5
(0, 0, 0, 0, 1)
21
1 + 2 = 3
d2
d3
d4
d5
(0, 1, 0, 0, 1)
19
2 + 10 = 12
(0, 0, 1, 0, 1)
16
2 + 3 = 5
(0, 0, 0, 1, 1)
11
2 + 2 = 4
(0, 0, 0, 0, 2)
1
2 + 1 = 3
d2
d3
d4
(0, 1, 0, 1, 1)
9
3 + 5 = 8
(0, 0, 1, 1, 1)
6
3 + 2 = 5
(0, 0, 0, 2, 1)
1
3 + 1 = 4
X
X

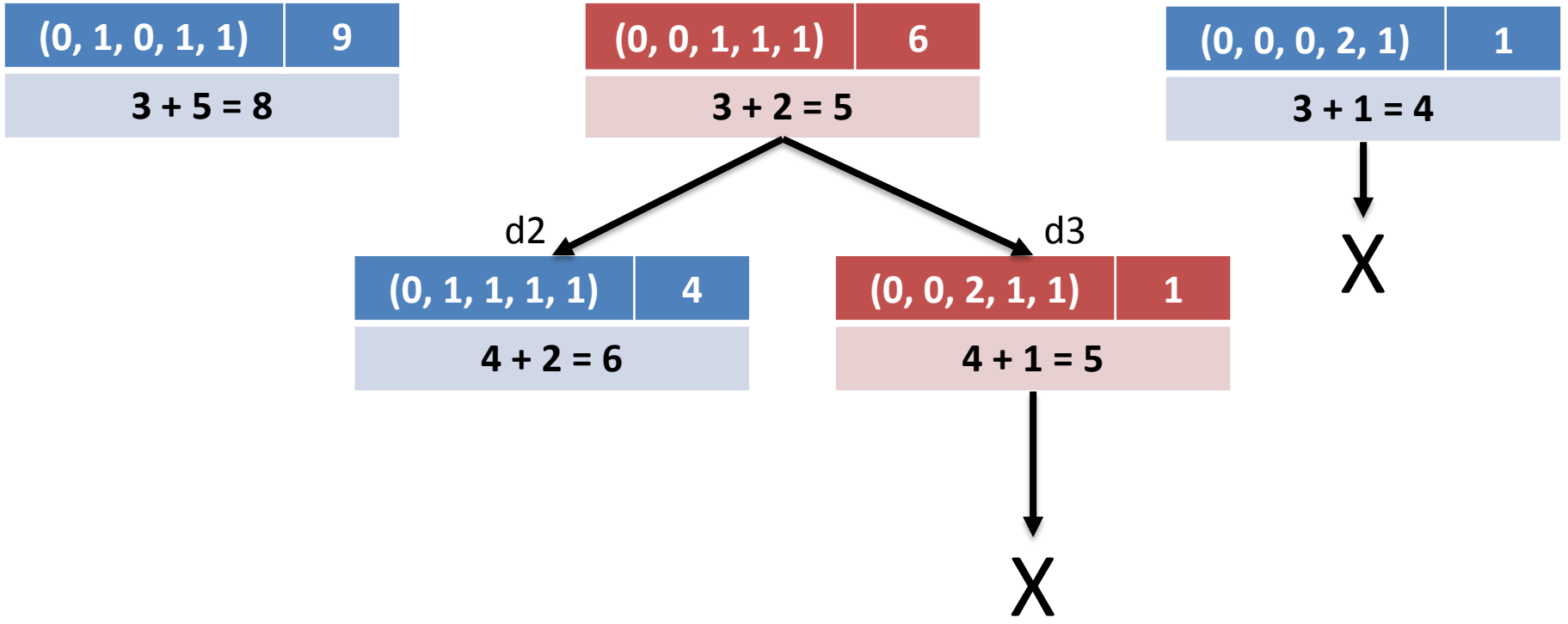
(0, 1, 0, 1, 1)	9
3 + 5 = 8	

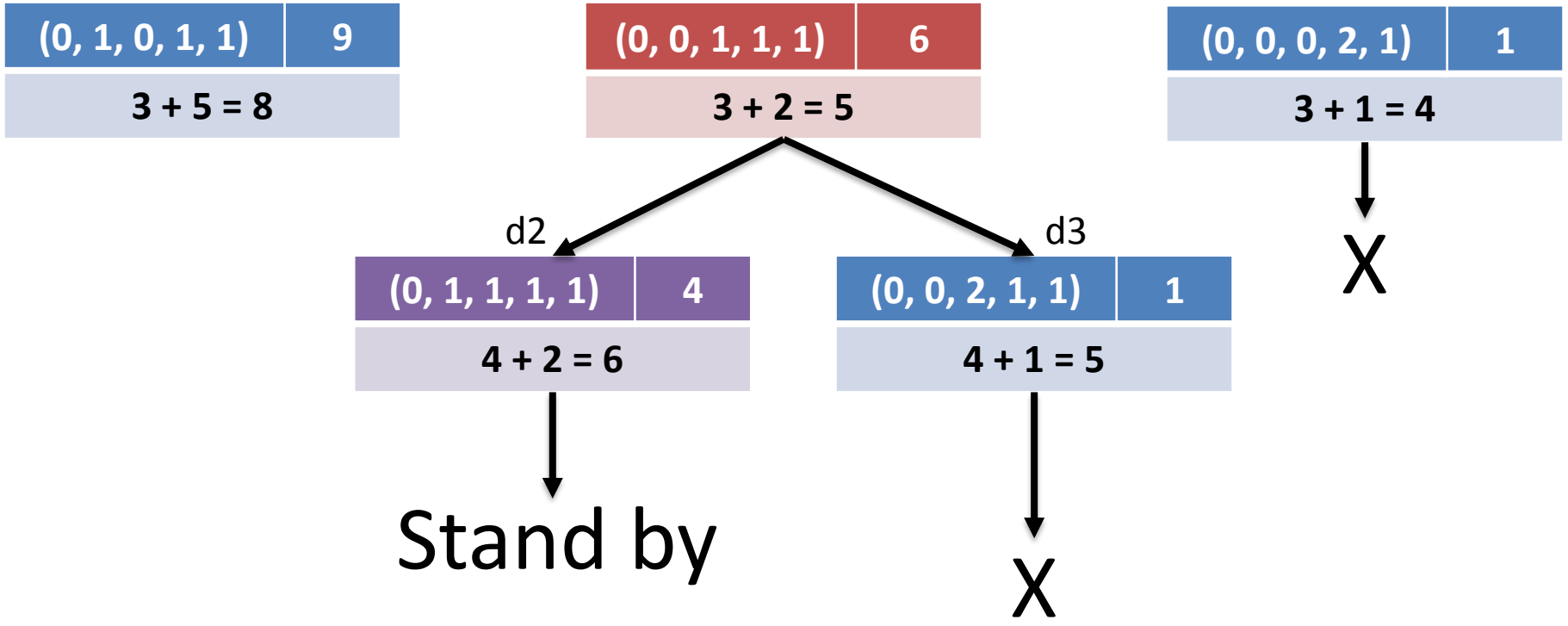
(0, 0, 1, 1, 1)	6
3 + 2 = 5	

(0, 0, 0, 2, 1)	1
3 + 1 = 4	

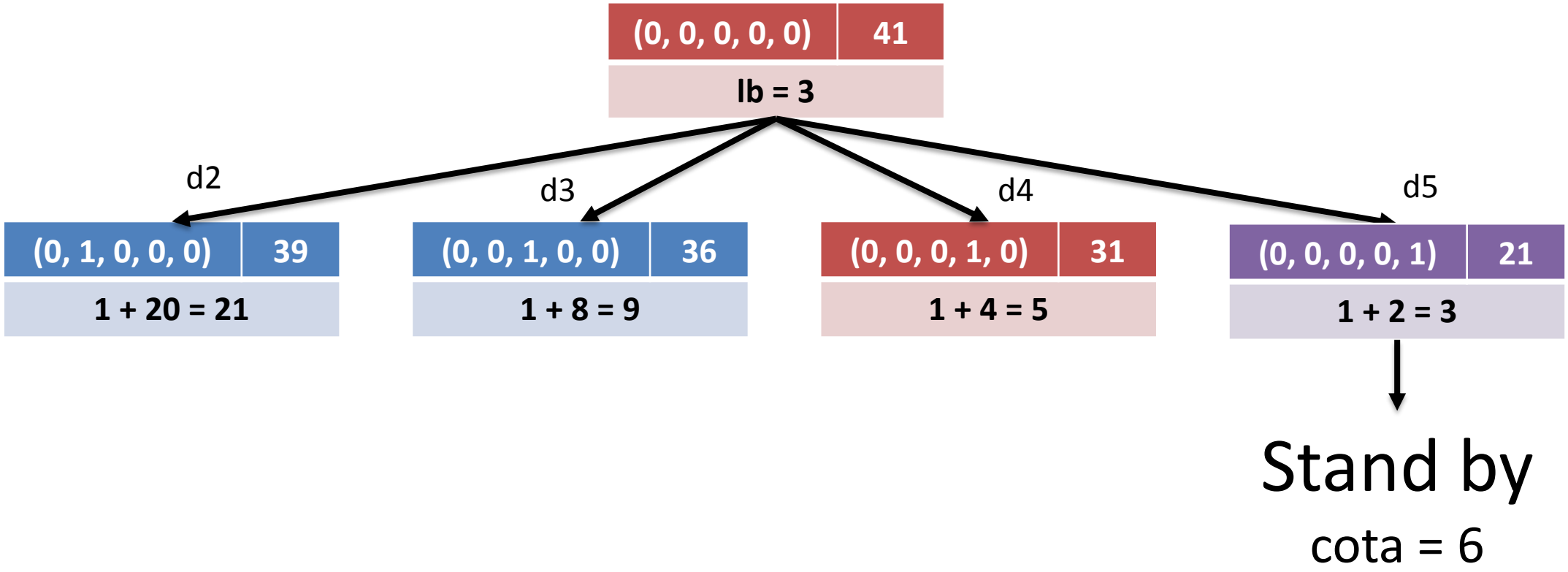
X

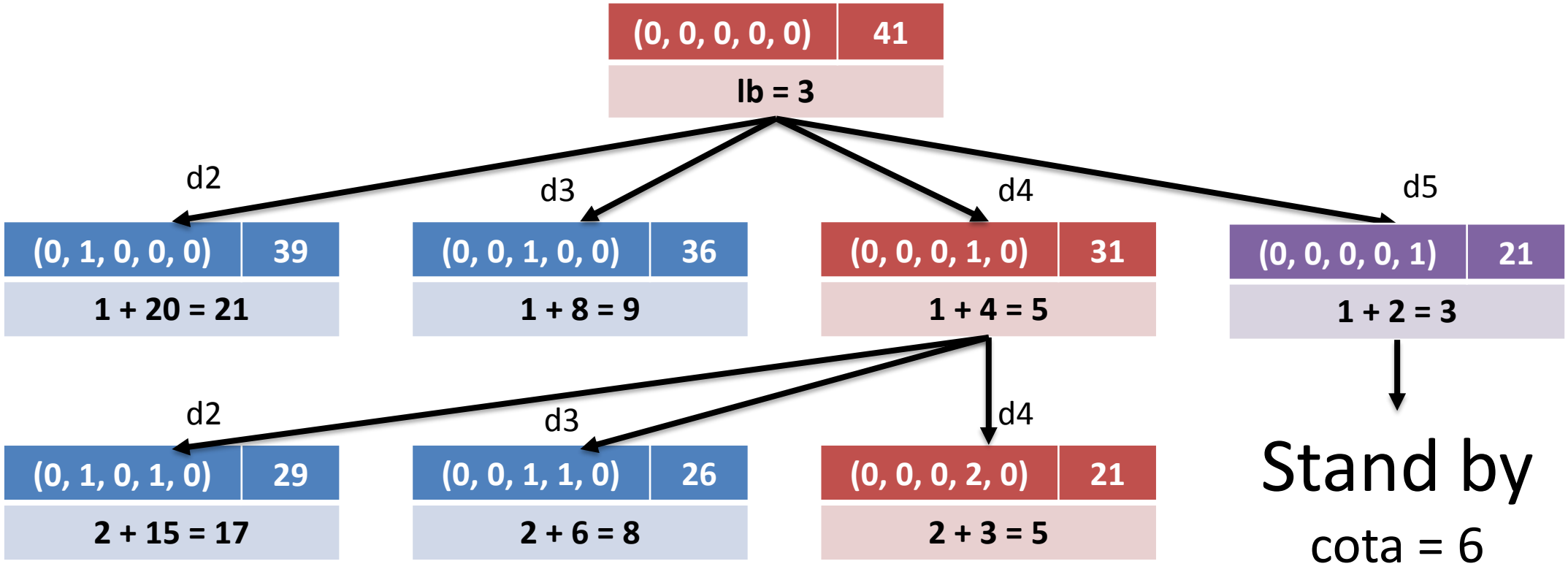

(0, 1, 0, 1, 1)
9
3 + 5 = 8
(0, 0, 1, 1, 1)
6
3 + 2 = 5
(0, 0, 0, 2, 1)
1
3 + 1 = 4
d2
d3
(0, 1, 1, 1, 1)
4
4 + 2 = 6
(0, 0, 2, 1, 1)
1
4 + 1 = 5
X

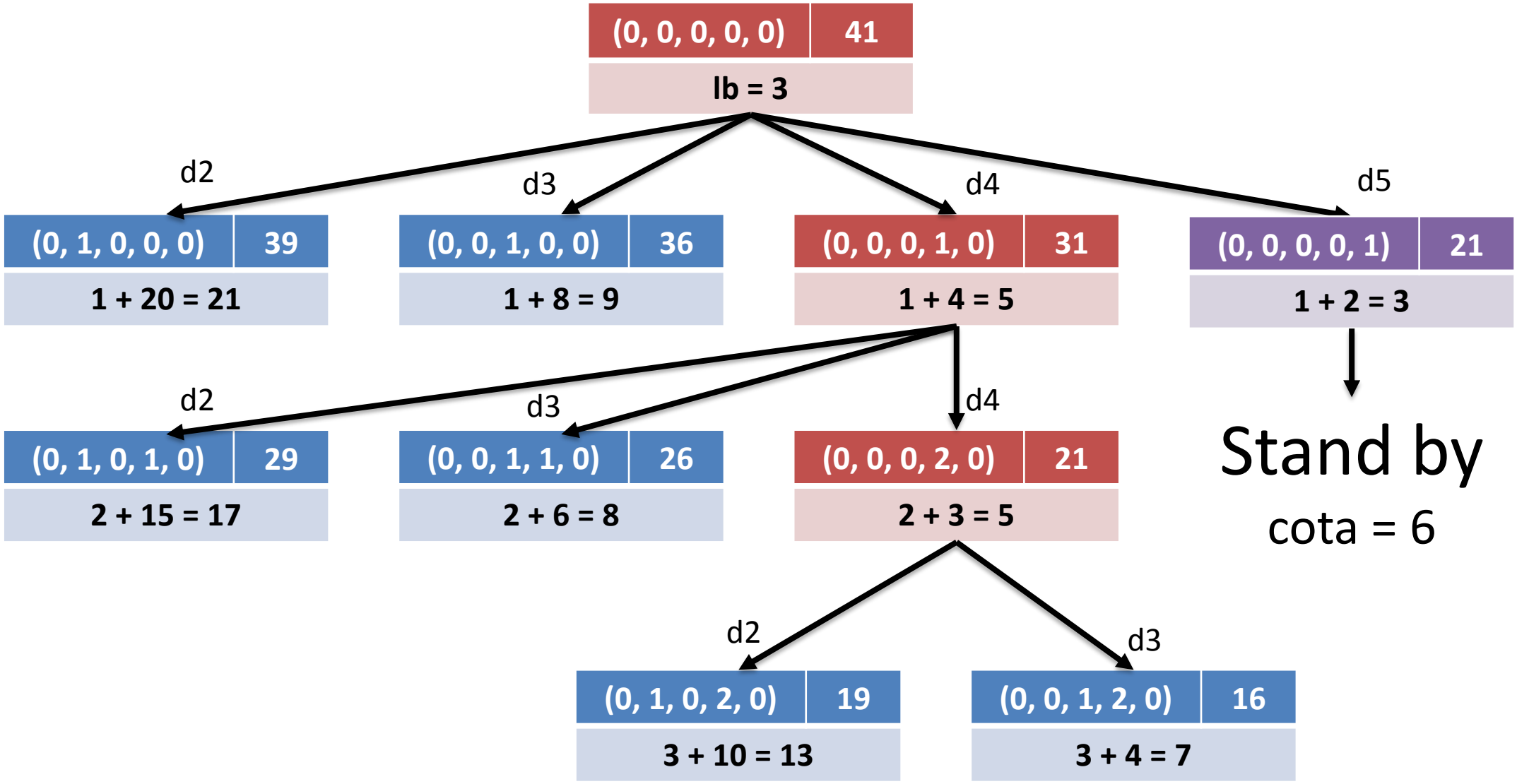

(0, 1, 0, 1, 1)
9
3 + 5 = 8
(0, 0, 1, 1, 1)
6
3 + 2 = 5
(0, 0, 0, 2, 1)
1
3 + 1 = 4
X
d2
d3
(0, 1, 1, 1, 1)
4
4 + 2 = 6
(0, 0, 2, 1, 1)
1
4 + 1 = 5
X



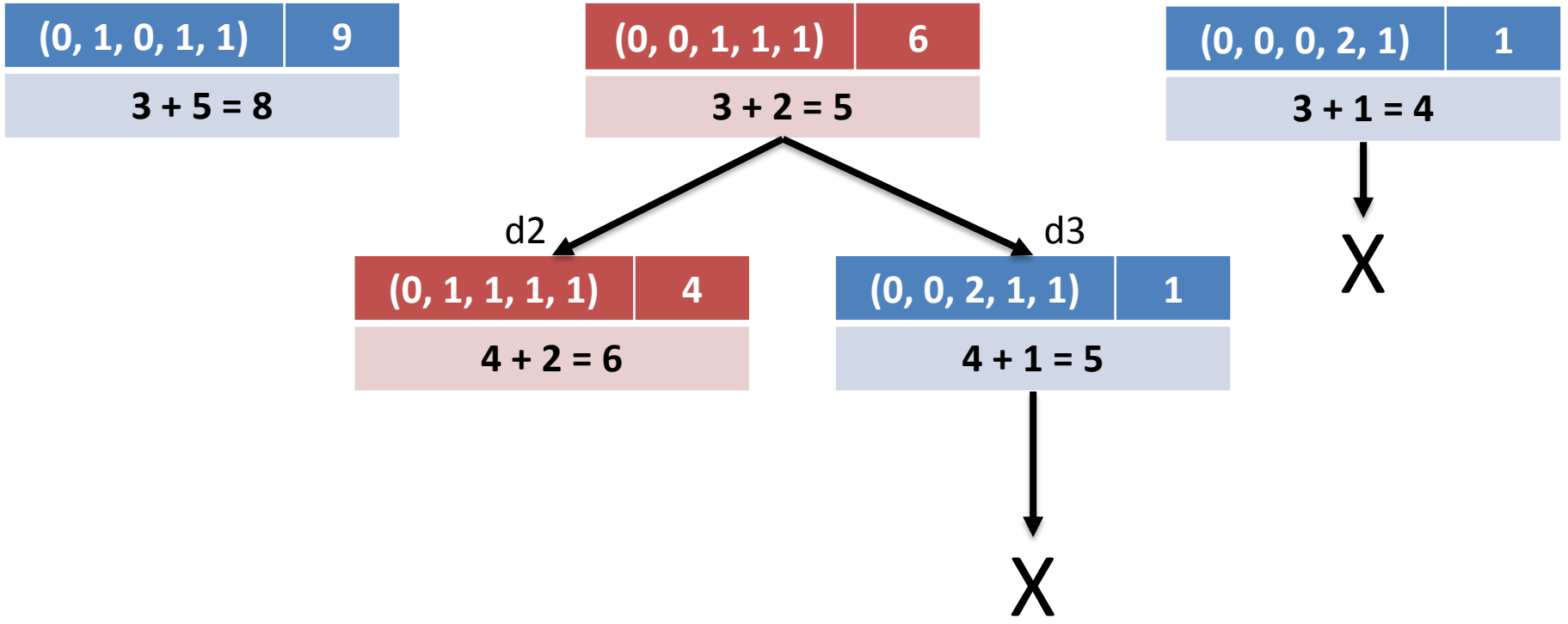
La cota de este nodo es 6, pero tenemos un nodo anterior cuya cota es de 5, por lo que este nodo anterior está antes en nuestra cola de prioridades o nuestro montículo (según la estruc. de datos utilizada), y tiene que ser visitado primero.

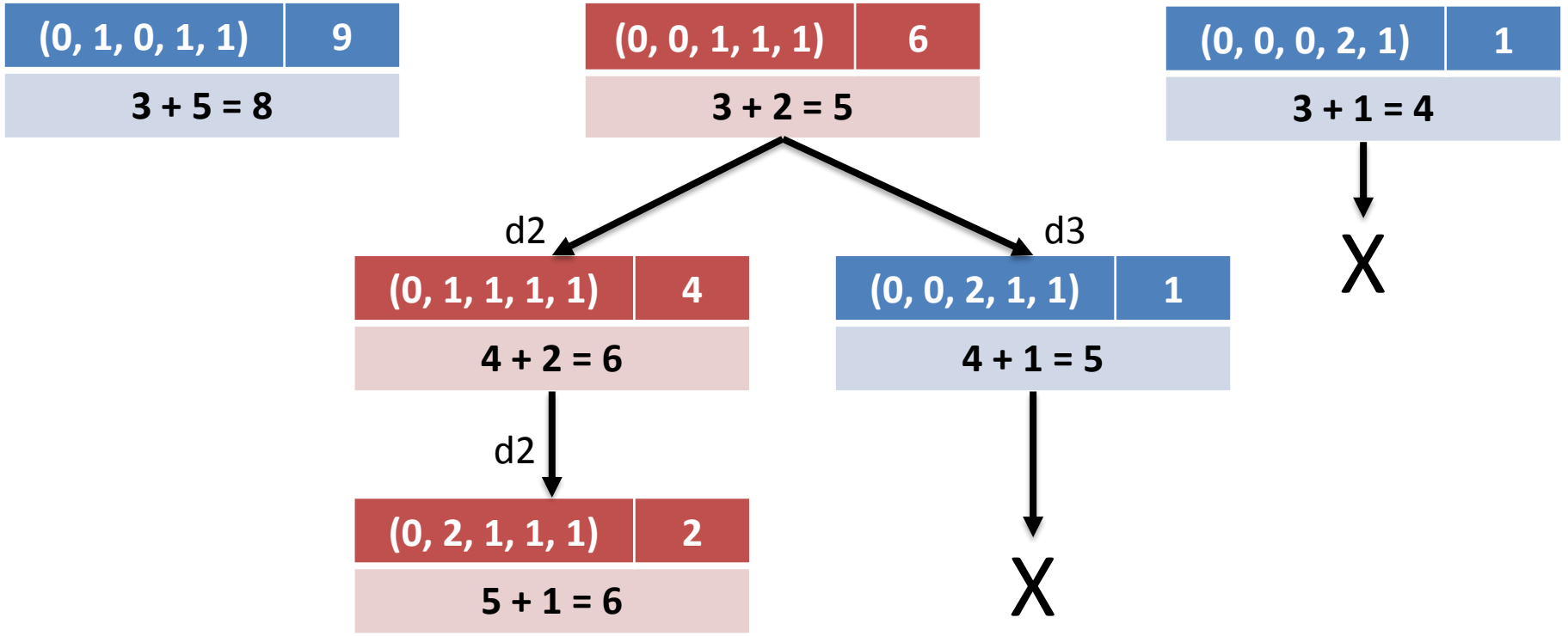

(0, 0, 0, 0, 0)
41
lb = 3
d2
d3
d4
d5
(0, 1, 0, 0, 0)
39
1 + 20 = 21
(0, 0, 1, 0, 0)
36
1 + 8 = 9
(0, 0, 0, 1, 0)
31
1 + 4 = 5
(0, 0, 0, 0, 1)
21
1 + 2 = 3
Stand by
cota = 6

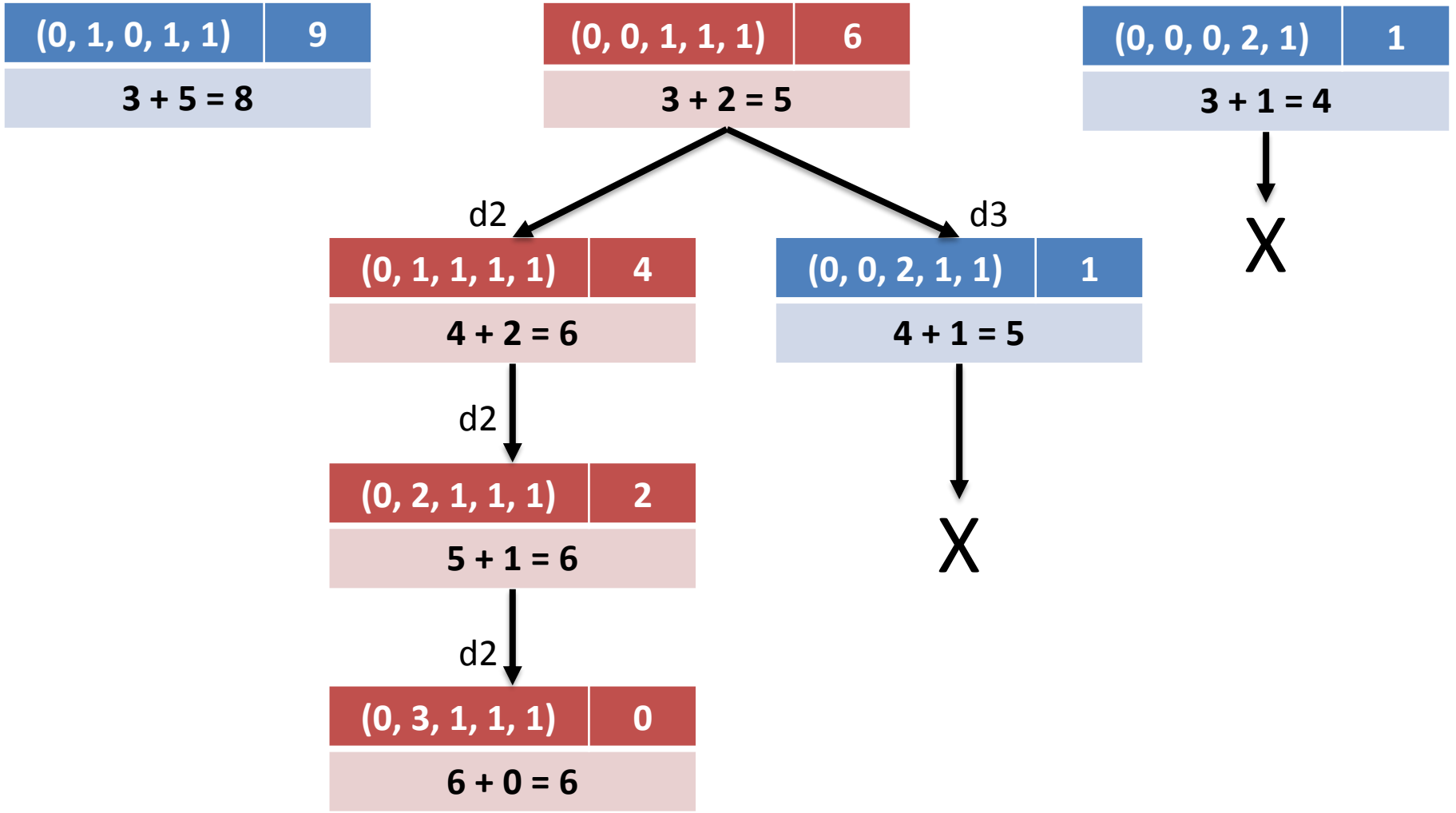

(0, 0, 0, 0, 0)
41
lb = 3
d2
d3
d4
d5
(0, 1, 0, 0, 0)
39
1 + 20 = 21
(0, 0, 1, 0, 0)
36
1 + 8 = 9
(0, 0, 0, 1, 0)
31
1 + 4 = 5
(0, 0, 0, 0, 1)
21
1 + 2 = 3
d2
d3
d4
(0, 1, 0, 1, 0)
29
2 + 15 = 17
(0, 0, 1, 1, 0)
26
2 + 6 = 8
(0, 0, 0, 2, 0)
21
2 + 3 = 5
Stand by
cota = 6

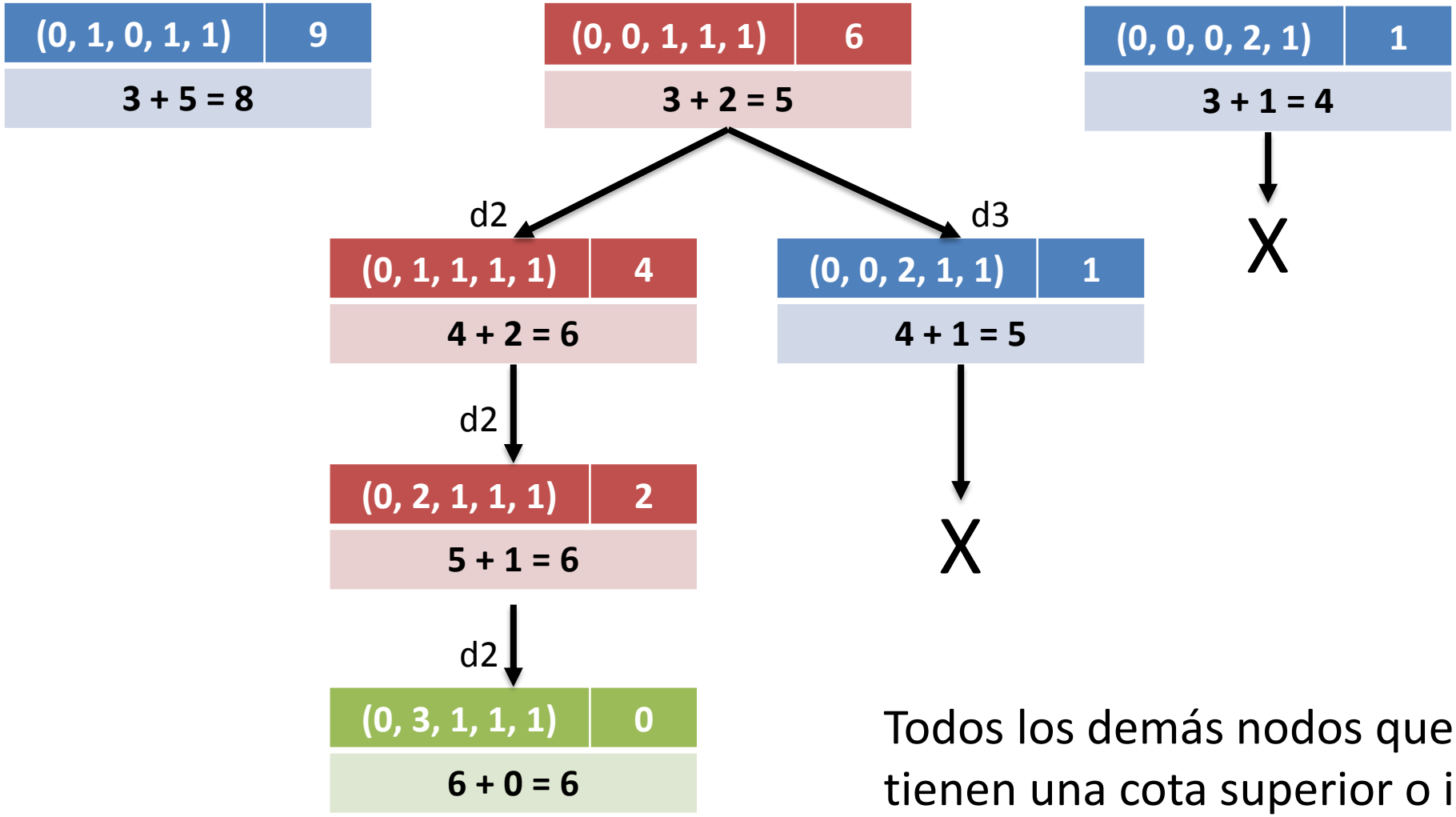


Volvemos al nodo en stand by cuya cota era 6.


(0, 1, 0, 1, 1)
9
3 + 5 = 8
(0, 0, 1, 1, 1)
6
3 + 2 = 5
(0, 0, 0, 2, 1)
1
3 + 1 = 4
X
d2
d3
(0, 1, 1, 1, 1)
4
4 + 2 = 6
(0, 0, 2, 1, 1)
1
4 + 1 = 5
X


(0, 1, 0, 1, 1)
9
3 + 5 = 8
(0, 0, 1, 1, 1)
6
3 + 2 = 5
(0, 0, 0, 2, 1)
1
3 + 1 = 4
X
d2
d3
(0, 1, 1, 1, 1)
4
4 + 2 = 6
(0, 0, 2, 1, 1)
1
4 + 1 = 5
d2
(0, 2, 1, 1, 1)
2
5 + 1 = 6
X

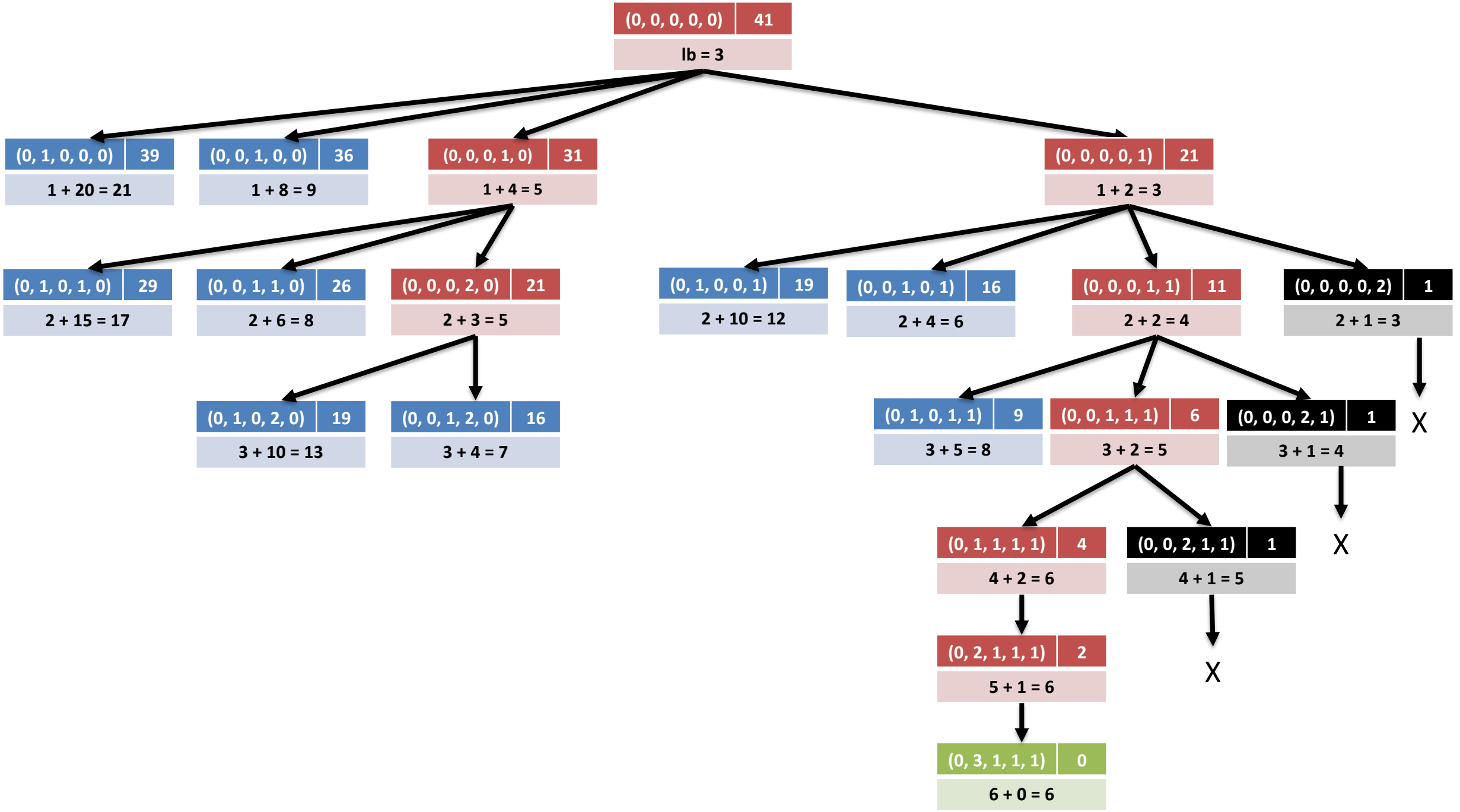

(0, 1, 0, 1, 1)
9
3 + 5 = 8
(0, 0, 1, 1, 1)
6
3 + 2 = 5
(0, 0, 0, 2, 1)
1
3 + 1 = 4
X
d2
d3
(0, 1, 1, 1, 1)
4
4 + 2 = 6
(0, 0, 2, 1, 1)
1
4 + 1 = 5
d2
(0, 2, 1, 1, 1)
2
5 + 1 = 6
X
d2
(0, 3, 1, 1, 1)
0
6 + 0 = 6



solución

Todos los demás nodos que quedan tienen una cota superior o igual a 6, por lo que deben podarse.

Esta solución es la óptima.


(0, 0, 0, 0, 0) 41
lb = 3
(0, 1, 0, 0, 0) 39
1 + 20 = 21
(0, 0, 1, 0, 0) 36
1 + 8 = 9
(0, 0, 0, 1, 0) 31
1 + 4 = 5
(0, 0, 0, 0, 1) 21
1 + 2 = 3
(0, 1, 0, 1, 0) 29
2 + 15 = 17
(0, 0, 1, 1, 0) 26
2 + 6 = 8
(0, 0, 0, 2, 0) 21
2 + 3 = 5
(0, 1, 0, 0, 1) 19
2 + 10 = 12
(0, 0, 1, 0, 1) 16
2 + 4 = 6
(0, 0, 0, 1, 1) 11
2 + 2 = 4
(0, 0, 0, 0, 2) 1
2 + 1 = 3
(0, 1, 0, 2, 0) 19
3 + 10 = 13
(0, 0, 1, 2, 0) 16
3 + 4 = 7
(0, 1, 0, 1, 1) 9
3 + 5 = 8
(0, 0, 1, 1, 1) 6
3 + 2 = 5
(0, 0, 0, 2, 1) 1
3 + 1 = 4
X
(0, 1, 1, 1, 1) 4
4 + 2 = 6
(0, 0, 2, 1, 1) 1
4 + 1 = 5
X
(0, 2, 1, 1, 1) 2
5 + 1 = 6
X
(0, 3, 1, 1, 1) 0
6 + 0 = 6

2. Se desea ensamblar n piezas de un vehículo de forma que el orden de ensamblaje de las mismas en las que se realiza influye en los costes debido a los movimientos que se ve obligado a realizar el brazo robótico. Para ello se tiene una matriz cuadrada C que nos proporciona los costes de ensamblaje, de forma que C_{ij} es el coste de ensamblar la pieza i después de haber ensamblado ya j piezas con $1 \leq i \leq n$ y $0 \leq j \leq n-1$. El objetivo de este problema es minimizar el coste de ensamblado de todas las piezas. Se pide:

   a) Escribir el esquema de vuelta atrás que mejor se adapta a este problema.
   b) Hacer una descripción detallada del árbol de búsqueda de la solución estableciendo:
      - Contenido general de un estado y contenidos del estado inicial, de un estado intermedio y de un estado final.
      - Esquema de ramificación de forma que no se repitan nodos.
      - Mostrar el árbol de expansión implícito para el ejemplo de este ejercicio utilizando Backtracking.
   c) Diseñar la(s) funciones de coste (cotas) necesarias para resolver este problema mediante la técnica de Ramificación y Poda. Mostrar el árbol de expansión implícito para el ejemplo de este ejercicio utilizando Ramificación y Poda.

a) Escribir el esquema de vuelta atrás que mejor se adapta a este problema.

```
Solucion backtracking (Solucion sol, Solucion msol, int mcal) {
    if (esSolucion(sol)) {
        int cal = calidad (sol);
        if (mcal < cal) return msol;
        else return sol;
    }
    else {
        Cont setCont = posContinuacion(sol);

        while (!esVacia(setCont)) {
            cont = selecciona(setCont);
            setCont = setCont – {cont};

            Solucion otra = backtracking(sol+cont, msol, mcal);
            if (calidad(otra) < mcal) {
                msol = otra; mcal =calidad(otra);
            }
        }
    }
}
```

El problema es de optimización.

Concretamente de minimización, por eso ponemos mcal < cal.

b) Hacer una descripción detallada del árbol de búsqueda de la solución estableciendo:
   - Contenido general de un estado y contenidos del estado inicial, de un estado intermedio y de un estado final.

Contenido de un estado:

- Vector con n componentes (tamaño n) y el coste asociado
  - En el vector V[j] = i significa que en el instante j se ensambla la pieza i.
  - El coste asociado es un número.

- Estado inicial: las n componentes están a 0 ($\forall j = 0, \dots, n-1 : V[j] = 0$); y el coste asociado es desconocido (¿?).

- Estado intermedio: algunas componentes desde la 0 hasta la k tienen valores distintos de 0 (se ha ensamblado una pieza) y ninguno se repite (las piezas son distintas); a partir del primer elemento igual a 0, todas las demás componentes son 0 (aún no se ha ensamblado nada). El coste es desconocido (¿?).

- Estado final: todas las componentes son distintas de 0 (se han ensamblado todas las piezas), no hay componentes repetidas y tendremos el coste asociado calculado para el proceso. Al final, al ser un problema de minimización, nos quedaremos con el estado final de menor coste asociado.

b) Hacer una descripción detallada del árbol de búsqueda de la solución estableciendo:
   - Esquema de ramificación de forma que no se repitan nodos.

- Niveles del árbol: en cada nivel se decide que pieza se ensambla en ese instante.
  - Nivel 1: que pieza se ensambla en el instante 0 (V[0]).
  - Nivel 2: que pieza se ensambla en el instante 1 (V[1]).
  - …
  - Nivel k+1: que pieza se ensambla en el instante k (V[k]).

- Ramificación: se puede ramificar por cada una de las piezas que aún no han sido ensambladas (por ejemplo, ordenándolas por número, de menor a mayor).

- De manera que en el nivel 1, se rellena la componente 0 (con los valores 1…n, de izquierda a derecha), con lo que la ramificación será de n.

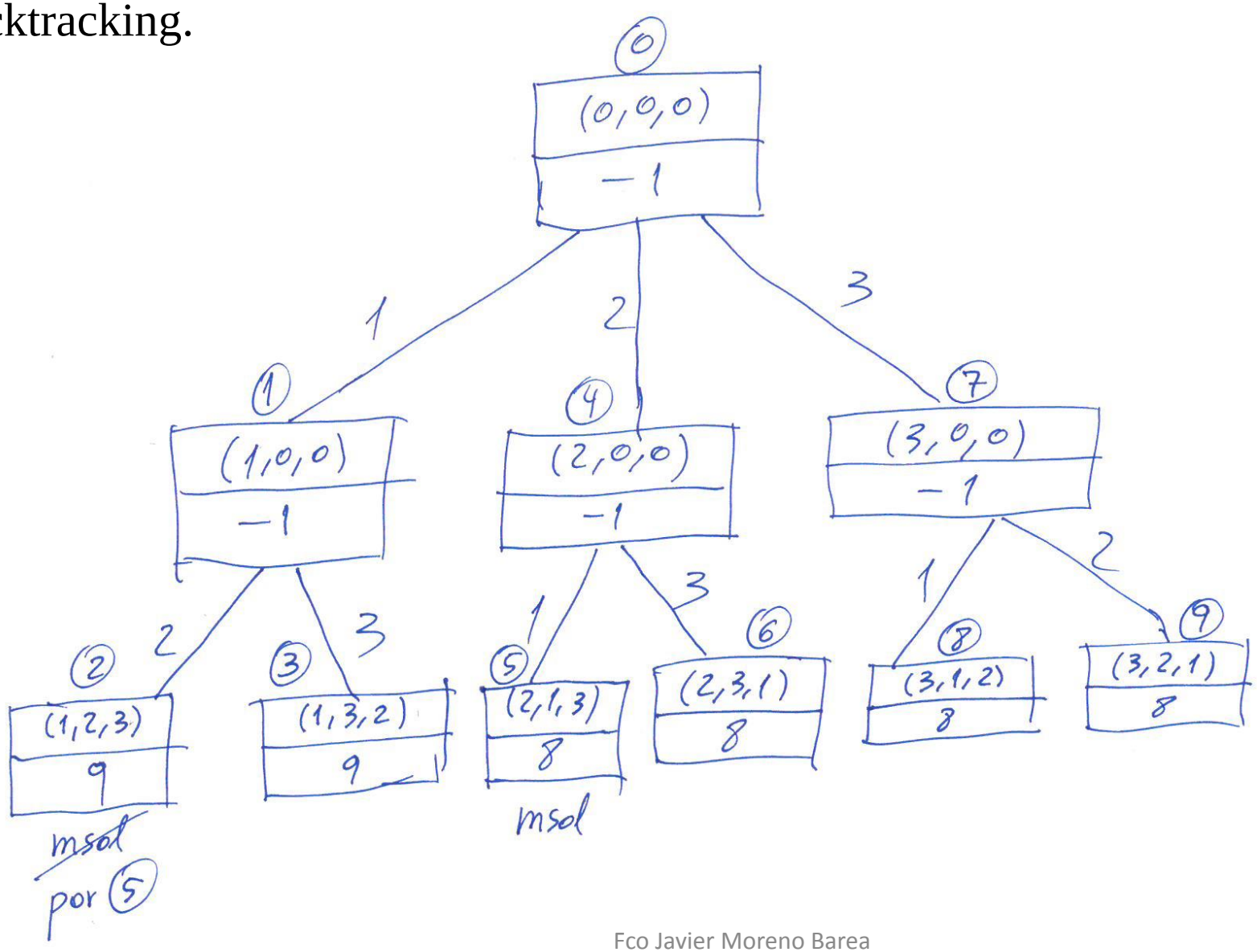
- En el nivel 2, se rellena la componente 1 y la ramificación es n-1, …, y en el nivel k+1 será n-k.

- En el nivel n-1, se rellena la componente n-1, y automáticamente, la componente n (solo queda esa pieza).

b) Hacer una descripción detallada del árbol de búsqueda de la solución estableciendo:

- Mostrar el árbol de expansión implícito para el ejemplo de este ejercicio utilizando Backtracking.

$$
\begin{array}{c} \\ i = \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \end{array}
\begin{array}{c} \begin{matrix} \quad & \mathrm{j} & \quad \end{matrix} \\ \begin{matrix} 0 & 1 & 2 \end{matrix} \\ \begin{bmatrix} 3 & 2 & 1 \\ 3 & 3 & 2 \\ 4 & 4 & 3 \end{bmatrix} \end{array}
$$



c) Diseñar la(s) funciones de coste (cotas) necesarias para resolver este problema mediante la técnica de Ramificación y Poda. Mostrar el árbol de expansión implícito para el ejemplo de este ejercicio utilizando Ramificación y Poda.

$$
\begin{array}{c} \\ i = \\ \begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \end{array}
\begin{array}{c}
\begin{matrix} \text{j} \\ 0 & 1 & 2 \end{matrix} \\
\begin{bmatrix} 3 & 2 & 1 \\ 3 & 3 & 2 \\ 4 & 4 & 3 \end{bmatrix}
\end{array}
$$

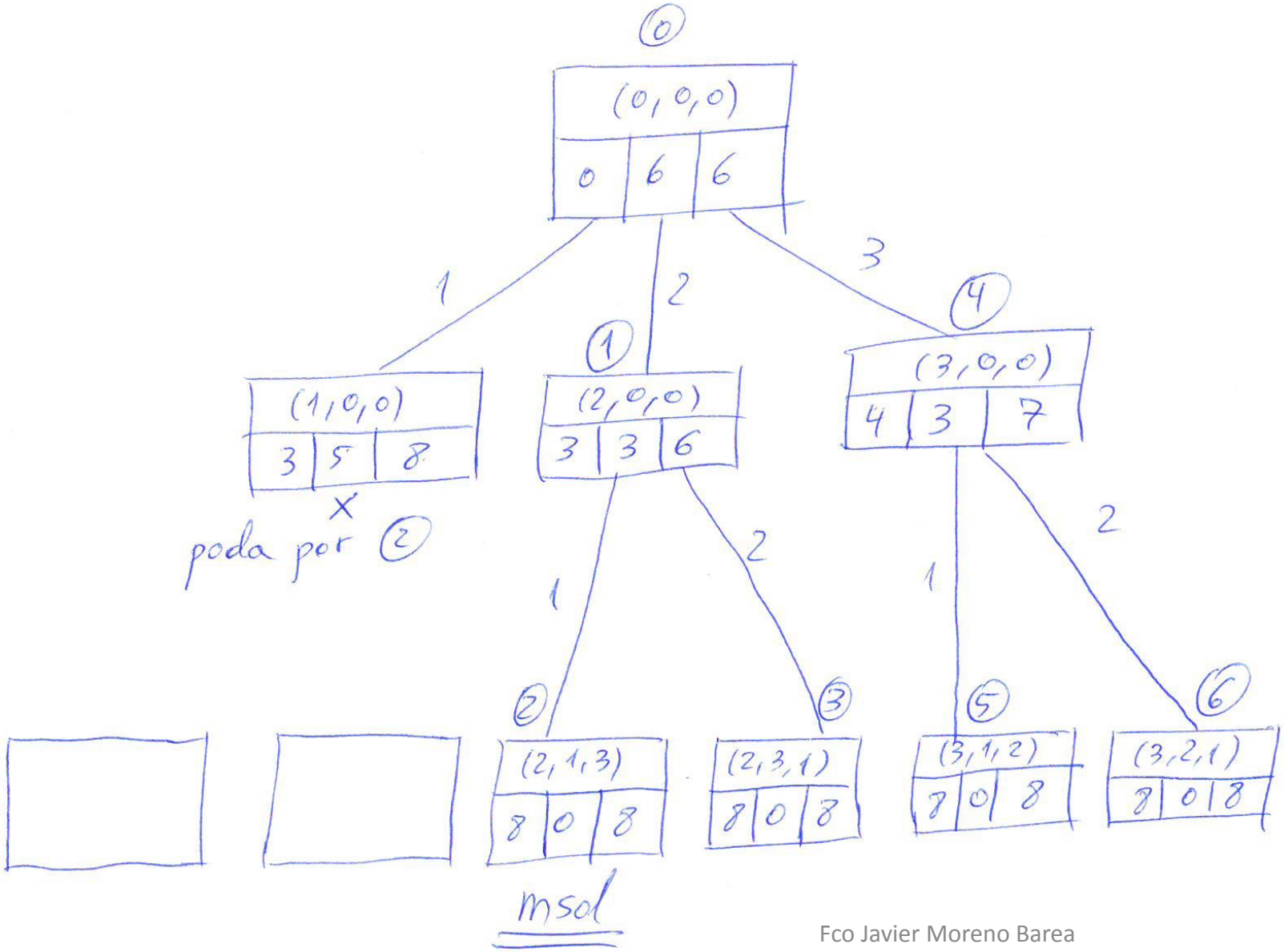
- Medida de calidad: f(x) = g(x) + h(x) donde,
  - g(x) = los costes de las asignaciones ya realizadas
  - h(x) = estimación optimista de mínimos

- En cada nodo tendremos el vector de n componentes donde V[j] = i significa que en el instante j se ensambla la pieza i; y las diversas medidas de calidad g, h, f.

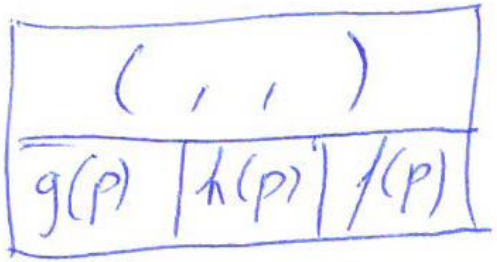
- Por ejemplo: nodo 0 (inicial) -> f(0) = g(0) + h(0).
  - Los costes de asignaciones ya realizadas es nulo, ya que no se ha realizado ninguno, g(0) = 0.
  - La estimación optimista de mínimos se calcula a partir de la selección de las piezas que dan el mínimo coste. Así, para el ejemplo, en posición j = 0 la pieza que da menor coste es 1 (coste 3), para j = 1 la pieza es 1 (coste 2), y para j = 2 la pieza es también 1 (coste 1). Así, h(0) = 3 + 2 + 1 = 6.

- En las siguientes ramificaciones, si habrán costes de asignaciones y las estimaciones se harán partiendo del coste ya producido, y sin tener en cuenta aquellas piezas que ya han sido asignadas.

c) Diseñar la(s) funciones de coste (cotas) necesarias para resolver este problema mediante la técnica de Ramificación y Poda. Mostrar el árbol de expansión implícito para el ejemplo.



$$i = \begin{array}{r} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} & j & \\ 0 & 1 & 2 \end{array} \\ \begin{bmatrix} 3 & 2 & 1 \\ 3 & 3 & 2 \\ 4 & 4 & 3 \end{bmatrix} \end{array}$$



[Ejercicio Voluntario] Una empresa se dedica a la traducción de textos entre varios idiomas. Para ello, la empresa dispone de algunos diccionarios, donde cada diccionario permite la traducción (bidireccional) entre dos idiomas. Sin embargo, no se dispone de diccionarios para cada par de idiomas existentes por lo que, normalmente, es preciso realizar varias traducciones entre diferentes idiomas para llegar a la traducción final. Concretamente, en la empresa se trabaja con N idiomas y M diccionarios.

Dados dos idiomas, se pide desarrollar un algoritmo que determine si es posible realizar la traducción y, en caso de ser posible, determine la cadena de traducciones de longitud mínima.

Se pide:

a) Hacer una descripción detallada del árbol de búsqueda de la solución estableciendo:
   - Contenido general de un estado, del estado inicial, de un estado intermedio y de un estado final.
   - Esquema de ramificación de forma que no se repitan nodos.

b) Escribir una implementación de vuelta atrás que se adapte a este problema.

c) Diseñar la(s) funciones de coste (cotas) necesarias para resolver este problema mediante la técnica de Ramificación y Poda, y determinar cuál es la complejidad del cálculo de la cota.

d) Simular la ejecución del algoritmo para un conjunto de datos sencillo propuesto por el estudiante.