```cpp
// Approccio bottom-up

/* La costante INF contiene un valore molto
 * alto, in modo che possa essere ignorato
 * quando viene chiamata la funzione min()
 */
const int MAXN = 5000, MAXK = 5000, INF = 1000000000;

// La matrice memo contiene le soluzioni dei sottoproblemi
int memo[MAXN + 1][MAXK + 1];

int mangia(int N, int K, int P[]) {
    for (int i = 1; i <= K; i++)
        memo[0][i] = INF;       // La prima riga della tabella viene settata a INF

    for (int i = 1; i <= N; i++) {      // Itera sul parametro i
        for (int j = 0; j <= K; j++) {  // Itera sul parametro j
            // Se scegliamo di non mangiare la i-esima portata
            memo[i][j] = memo[i - 1][j];

            // Se scegliamo di mangiarla
            if (P[i - 1] <= j)  // Se j - P[i - 1] non e' negativo
                /* Al peso della i-esima portata (P[i - 1]) sommiamo la soluzione
                 * ottima contenuta in memo[i - 1][j - P[i - 1]], e restituiamo il
                 * minimo tra i due candidati
                 */
                memo[i][j] = min(memo[i][j], memo[i - 1][j - P[i - 1]] + P[i - 1]);
            else                // Altrimenti
                /* Come prima, ma - dato che j - P[i - 1] e' negativo - basta
                 * mangiare la i-esima portata, senza sommare nient'altro
                 */
                memo[i][j] = min(memo[i][j], P[i - 1]);
        }
    }

    // Restituisce la soluzione del problema
    return memo[N][K];
}
```