Problem $Pm||C_{max}$: dany jest zbiór $M = \{1, \dots, i, \dots, m\}$ m równoległych, identycznych maszyn oraz zbiór $J = \{1, \dots, j, \dots, n\}$ n niepodzielnych zadań. Każde z zadań charakteryzuje się czasem wykonywania p_j. Zadanie może być wykonywane dokładnie przez jedną maszynę. Należy tak przyporządkować zadania do maszyn, aby czas zakończenia wykonywania ostatniego z zadań (tego, które zakończy się jako ostatnie) był jak najmniejszy, zatem nasze $C_{max} = \max_{j \in J} C_j$.

Ocena	Zadania
3.0	Algorytmy LSA, LPT
3.5	PD dla $P2\|\|C_{max}$
4.0	Przegląd zupełny dla $P2\|\|C_{max}$
4.5	PTAS dla $P2\|\|C_{max}$ FPTAS dla $P2\|\|C_{max}$
5.0	PTAS dla $P3\|\|C_{max}$, przegląd zupełny dla $P3\|\|C_{max}$, PD dla $P3\|\|C_{max}$

Na ocenę wyższą niż 3.0 należy wykonać też wszystkie zadania na niższą ocenę, np. na ocenę 4.0 należy wykonać też zadania na ocenę 3.5 oraz 3.0

Programowanie dynamiczne: https://rtime.ciirc.cvut.cz/~hanzalek/KO/sched_e.pdf

Programowanie dynamiczne dla P2||Cmax (szkic):

Alokujemy tablicę *T* o rozmiarze n+1 wierszy i Kl=$(\sum_{j=1}^{n} p_j)/2 + 1$ kolumn (bierzemy liczbę całkowitą, podłogę). To +1 jest dlatego, że dodajemy dla ułatwienia sztuczne zadanie 0 o zerowym czasie wykonywania. Uzupełniamy całą tablicę na przykład zerami. Całą pierwszą kolumnę uzupełniamy 1. I teraz po kolei, dla każdego zadania (idziemy wierszami) uzupełniamy:

```
for(j=1; j<=n; j++)
        for(k=1; k<Kl; k++)
                if (T[j-1][k]==1)||((k>=pj)&&(T[j-1][k-pj]==1))
                        T[j][k]=1
```

Następnie należy zrobić backtracking, aby „odszyfrować” rozwiązanie.

Uwaga, jeżeli w powyższym kodzie jest błąd, to proszę o informację.