AAL - Analiza algorytmów
Koncepcja rozwiązania zadania projektowego

1. **Treść:**
   Dany jest zbiór sal wykładowych oraz zbiór zamówień, określających czas rozpoczęcia i zakończenia wykładu. Ułożyć plan wykorzystania sal, akceptując pewne wykłady i odrzucając inne, tak aby sumaryczny czas wykorzystania sal był jak najdłuższy. Porównać czas obliczeń i wyniki różnych metod.

2. **Założenia:**
   a. ***Sale*** - podane w pliku tekstowym (|| podane na wejście || w konsoli || generowana losowa ilość), każda w nowej linii reprezentowana ciągiem znaków jako identyfikator (nie ma to większego znaczenia, tylko do celów wyświetlenia wyników działania algorytmów),
   b. ***Zamówienia*** - podane w pliku tekstowym (|| generowane || ...) jako sekwencja trzech liczb całkowitych: pierwsza reprezentująca dzień tygodnia, druga godzinę rozpoczęcia, trzecia godzinę zakończenia wykładu. Dzień tygodnia interpretowany jest jako *'początek||koniec += dzień_tygodnia * 24'* co daje liniowy układ czasu.

3. **Struktury danych w modelu:**
   a. Lista || wektor, *zamówień*
   b. Lista || wektor, *sal*
   c. Struktura składająca się z wektora sal przechowujących listy kolejno przypisywanych wykładów - w celach czysto wynikowej prezentacji planu zajęć.

4. **Algorytmy:**
   a. Algorytm programowania dynamicznego, iteracyjnie dla każdej sali, znajdujący lokalnie optymalny układ dla aktualnie rozważanej sali z wykładów nie umieszczonych w salach poprzednich. Działa optymalnie dla liczby sal = 1, można znaleźć przypadki, że algorytm nie znajdzie optymalnego rozwiązania problemu, aczkolwiek działa szybko. Sortujemy zajęcia wg. czasu zakończenia, następnie dla kolejnych wykładów sprawdzamy optimum czasowe gdybyśmy przypisali wykład do rozważanej sali. W tym celu należy porównać dwie liczby *a* i *b*, gdzie *a* jest czasem trwania rozważanego wykładu, *b* jest sumą czasu trwania rozważanego wykładu i największego wykorzystania sali uzyskanego po dołączeniu wcześniejszych wykładów, który znajdujemy spośród wykładów kończących się nie później niż w chwili rozpoczęcia rozważanego wykładu. Ułożenie zajęć w następnych salach następuje z wyłączeniem już wstawionych zajęć do sal poprzedzających.

```
sort(wyklady);
for (s: sale)
        for (w: wyklady) {
                t1 = czas_trwania(w);
                t2 = t1 + dotychczasowe_najlepsze_
                        wykorzystanie_sali_przez_wyklady_
                        konczace_sie_nie_pozniej_niz_
                        czas_w.poczatek

                optimum_czasu_dla_w = max(t1, t2);
        }
```

   b. Algorytm zachłanny, wykonywany równolegle dla wszystkich sal. Zajęcia posortowane wg. czasu zakończenia. Następnie iteracyjnie przechodząc po kolejnych

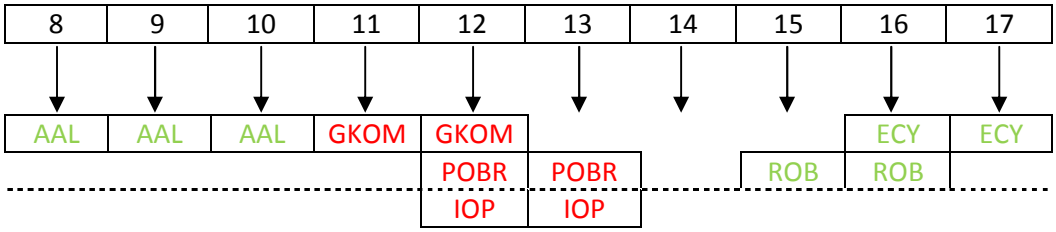
wykładach dopasowanie do sali nr. 1, jak nie pasuje (nakłada się) to do sali nr. 2 itd. Nie gwarantuje to optymalnego układu, aczkolwiek zapewnia szybkie działanie.

```
sort(wyklady);
for (w: wyklady)
	for (s: sale)
		if (nie nakłada się)
			umieść w konkretnej Sali
	// jak nie pasuje nigdzie => odrzuć
```

c. *Brute Force*, sprawdzenie wszystkich możliwych kombinacji i wybranie najlepszej możliwej. Rekurencyjnie sprawdzamy kolejne możliwości ulokowania zajęć, odrzucając na wstępie kombinacje z nakładającymi się zajęciami, co zmniejsza ilość możliwości. Dla dużych ilości wykładów + sal narzut czasowy obliczeń jest ogromny.

```
Ulozenie {
	wektor_sal;
};

kombinacje(zajecia, od_ktorego, ulozenie, ilosc_sal) {
	if (ilosc_sal < 1)
		return;

	for (i = od_ktorego; i < zajecia.size) {
		nowe_ulozenie = ulozenie;
		nowe_ulozenie[ulozenie.size - ilosc_sal] = zajecia[i];

		if (wystepuje nakladanie w nowym ulozeniu)
			return; // szukamy nie nakladajacych sie

		// Rozwazone zajecie wykluczone z dalszych rozwazan
		nowe_zajecia = zajecia - zajecia[i];

		// Rozwazenie permutacji w aktualnej sali
		kombinacje(nowe_zajecia, i, nowe_ulozenie, sal_ile);
		// Rozwazenie permutacji w nastepnej sali
		kombinacje(nowe_zajecia, 0, nowe_ulozenie, sal_ile - 1);

	}
}
```

d. Połączenie algorytmów używanych na mniejszych pod-problemach. Tworzymy wektor odpowiadający kolejnym godzinom, do każdej godziny tworzymy listę do której podpinamy informację o wykładzie odbywającym się w tym czasie. Jeżeli maksymalna długość listy jest mniejsza równa ilości dostępnych sal, to można przyporządkować sale po kolei. Jeżeli nie to dla skupisk nakładających się sal w miejscach gdzie nakłada się więcej zajęć niż jest sal wykonujemy jeden z wyżej wymienionych algorytmów w celu znalezienia odpowiedniego układu. Zajęcia nie nakładające się po prostu przypisujemy do sal. Algorytm w niektórych przypadkach łańcuchowego nakładania się wykładów, niekiedy sprowadza się do działania jednego w ww. podejść.



np. gdy dostępne 2 sale