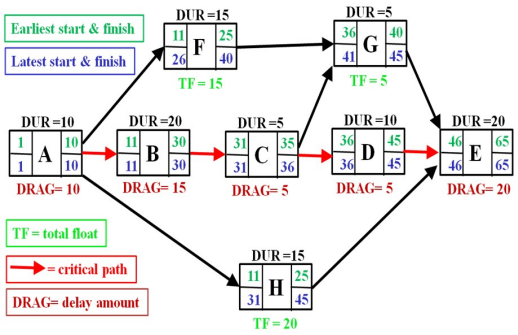
CPM - Critical Path Method



Bildquelle:
https://en.wikipedia.org/wiki/Critical_path_method
Vgl. auch:
https://de.wikipedia.org/wiki/Netzplantechnik

Betrachtet werden unterschiedliche Aspekte der Entwicklung einer Software zur Unterstützung von Projektplanungen in der Bauindustrie. Dabei soll es sich um das Entwickeln einer Netzplan-SW handeln.

Das zu entwickelnde Software-Tool soll Hilfestellung für eine möglichst zeitoptimale Durchführung eines Projektes. Es wird sich an der Critical-Path-Method (CPM) orientiert. Die einzelnen Prozesse sind zum Teil von einander abhängig. Jeder Prozess besitzt eine ID, einen Namen und eine Ausführungsdauer. Außerdem soll für jeden Prozess der früheste Anfangszeitpunkt (FAZ, der späteste Anfangszeitpunkt (SAZ), der früheste Endezeitpunkt (FEZ) und der späteste Endezeitpunt (SEZ) verwaltet werden.

1. Aufgabe – Use-Case-Diagramm

Betrachtet werden kann das Anwendungsfalldiagramm unter dem Aspekt zur Unterstützung von Projektplanungen in der Bauindustrie. (Dort lässt es sich tatsächlich sehr gut einsetzen.)

Die Prozesse und deren Abhängigkeiten werden von dem leitenden Bauingenieur festgelegt und benannt. Die einzelne Prozesslängen werden vom leitenden Bauingenieur abgeschätzt und in das dem System eingegeben.
Während der Projektdurchführung ist es nicht ungewöhnlich, dass die angesetzte Prozessdauer aktualisiert (verändert) werden muss. Diese Aktualisierungen werden ebenfalls vom leitenden Bauingenieur in das System eingepflegt.
Die Terminplanung, also die Berechnung des kritischen Pfades, die Berechnung der FAZ/FEZ und die Berechnung der SAZ/SEZ kann von dem Geschäftsführer oder dem leitenden Bauingenieur angestoßen werden.

Erstellen Sie ein entsprechendes Use-Case-Diagramm.

2. Minimum- und Maximumberechnung (Struktogramme und Quelltexterstellung)

Für die Berechnung der FAZ bzw. SEZ eines Prozesses ist die Suche des Minimums bzw. des Maximums der entsprechenden Zeitpunkte der bedingenden Prozesse notwendig. (Der FAZ setzt sich als das Maximum über alle FEZ der im Voraus abzuschließenden Prozesse fest; der SEZ setzt sich als das Minimum über alle SAZ aller von diesem Prozess abhängigen Prozessen fest.)

Erstellen Sie ein Struktogramm, welches aus einem Array das Maximum, den Index der Maximumposition, das Minimum und den Index der Minimumposition bestimmt und ausgibt.

3. ProjektPlanungsTool - ProPlan

Für die erste SW-Version soll eine Klasse **`Process`** konzipiert werden, die die einzelnen Prozesse bzw. Vorgänge verwaltet. Neben den oben aufgeführten Attributen sollen die Vorgänger- und die Nachfolger-Prozesse eines Vorgangs als Attribute verwaltet werden.

a. Erstellen Sie diese Klasse in JAVA. Sie können davon ausgehen, dass ein Vorgang maximal 10 Vorgänger- und maximal 10 Nachfolger- Prozesse besitzt.
Versehen Sie diese Klasse mit einem Konstruktor mit dem sich die Attribute ID, der Namen und die Ausführungsdauer initialisieren lassen. Die Attribute FAZ, FEZ, SAZ, SEZ, die Arrays bzw. Listen der Vorgänger und Nachfolger sollen auf Werte initialisiert werden, die als ungültig zu erkennen sind (z.B. **`null`** oder **`-1`**).

b. Erweitern Sie die Klasse um folgende getter- und setter- Methoden:

```
int   getID()
String      getName()
int   getDauer()

void  setID  (int id)
void  setName (String name)
void  setDauer (int dauer)
```

c. Erweitern Sie die Klasse um eine Methode die es ermöglicht einen Vorgang als Vorgängerprozess in die Vorgänger-Liste/Array eines Prozesses einzutragen. Verwenden Sie für die anzulegende Methode folgende Signatur.

```
void  addPredecessor (Process preProcess);
```

Im Folgenden können Sie davon ausgehen, dass es ebenfalls eine Methode gibt, die nachfolgende Prozesse entsprechend einträgt. Diese Methode besitzt folgende Signatur:

```
void  addSuccessor (Process succProcess);
```

d. Erweitern Sie die Klasse um eine Kontrollmethode: Diese soll sicherstellen, dass ein als Nachfolger einzutragender Prozess nicht schon als (direkter) Vorgänger eingetragen ist.
Damit sollen unauflösbare Abhängigkeiten vermieden werden. Diese Methode besitzt folgende Signatur:

```
boolean isPredecessor (Process process)
```

e. Erweitern Sie die Klasse um eine Methode zum abgesicherten Eintragen der Nachfolger.
Nur wenn diese noch nicht als Vorgänger eingetragen sind, soll die entsprechende Eintragung erfolgen.

f. Erweitern Sie die Klasse um Methoden, die angeben ob es sich bei dem Prozess um den Start- oder um den Ende-Prozess handelt. Als Entscheidungskriterium können Sie die Anzahl der Vorgänger bzw. der Nachfolger heranziehen. Folgende Signatur ist vorgesehen:

```
boolean isStartProcess ()
boolean isEndProcess ()
```

g. Erweitern Sie die Klasse um eine Methode die den FAZ eines Prozesses bestimmt und den Wert als Attribut einsetzt. Falls es sich um den Anfangsprozess handelt, soll der FAZ mit 0 angesetzt werden. Diese Methode besitzt folgende Signatur:

```
void calcFAZ ()
```

Im Folgenden können Sie davon ausgehen, dass es ebenfalls eine Methode gibt, die den SEZ eines Prozesses berechnet. Diese Methode besitzt folgende Signatur:

```
void calcSEZ ()
```

h. Erweitern Sie die Klasse um eine Methode die ermittelt, ob der Prozess kritisch ist.
(Ist der Prozess Teil des kritischen Pfades). Das ist der Fall, wenn der Gesamtpuffer zu Null wird.
Der Gesamtpuffer errechnet sich aus GP:= SAZ - FAZ, also muss für den kritischen Pfad gelten:
FAZ = SAZ
(Hinweis: Der SAZ eines Prozesses lässt sich ermitteln durch: SAZ:= SEZ - Dauer)
Diese Methode besitzt folgende Signatur:

```
boolean isCritical ()
```

i. Erstellen Sie eine JAVA-Routine, die bei den vorgegebenen Prozessen und deren Abhängigkeiten überprüft, ob es sich bei dem Prozess mit der id = 3; name = "B2" um einen kritischen Prozess handelt. Setzen Sie dazu die oben angegebenen Klassen-Methoden ein.

id	name	dauer	ID der Vorgänger Predecessors	ID der Nachfolger Successors
0	Start	0	-	1
1	A1	1	0	2; 3
2	B1	1	1	4
3	B2	3	1	4
4	End	0	2; 3	-

4. ProjektPlanungsTool - ProPlan

Für eine weitere SW-Version soll eine Klasse **`ProcessXT`** erstellt werden, die von der Klasse **`Process`** erbt. Diese neue Klasse soll über Attribute und Methoden verfügen, die die Pufferzeiten *Gesamtpuffer* und *Freier Puffer* berechnet. Diese Pufferdaten sollen dann zum Ermitteln des *kritischen Pfades* herangezogen werden.