Билет № 21

1. **Сформулювати принцип оптимальності Беллмана**. БЕЛЛМАНА ПРИНЦИП ОПТИМАЛЬНОСТИ [Bellman's optimality principle] — важнейшее положение динамического программирования, которое гласит: оптимальное поведение в задачах динамического программирования обладает тем свойством, что каковы бы ни были первоначальное состояние и решение (т. е. "управление"), последующие решения должны составлять оптимальное поведение относительно состояния, получающегося в результате первого решения. Этот принцип можно выразить и рассуждая от противного: если не использовать наилучшим образом то, чем мы располагаем сейчас, то и в дальнейшем не удастся наилучшим образом распорядиться тем, что мы могли бы иметь.Следовательно, если имеется оптимальная траектория, то и любой ее участок представляет собой оптимальную траекторию. Этот принцип позволяет сформулировать эффективный метод решения широкого класса многошаговых задач. (Подробнее см. Динамическое программирование.)

2. **Коли застосовують евристичні алгоритми**? В інформатиці, евристичний алгоритм, або просто евристика, це алгоритм спроможний видати прийнятне рішення проблеми серед багатьох рішень, але неспроможний гарантувати, що це рішення буде найкращим. Отже такі алгоритми є приблизними і неточними. Зазвичай такі алгоритми знаходять рішення близьке до найкращого і роблять це швидко. Часто, можна знайти таку задачу, що евристичний алгоритм буде працювати або дуже довго, або видавати невірні результати, однак, такі парадоксальні приклади можуть ніколи не зустрітись на практиці через свою специфічну структуру. Таким чином, використання евристики дуже поширене в реальному світі. Для багатьох практичних проблем евристичні алгоритми, можливо, єдиний шлях для отримання гарного рішення в прийнятний проміжок часу. Існує клас евристичних стратегій названих метаалгоритмами, котрі часто використовують, наприклад, випадковий пошук. Такі алгоритми можуть бути застосовані до широкого кола завдань, при цьому хороші характеристики не гарантуються.

3. **Яка програма виконує запис інформації в область збереження та що там зберігається.** Ядро ОС. Записываются значения регистров при переключении процесов из состояния в состояние либо их синхронизации.

4. **Що таке резидентні вершини та як їх знайти? Навіщо вони потрібні?** Вершини, що увійшли в критичний шлях, а їх часові та просторові координати не повинні змінюватись при вирішенні задач. Знайти критичний шлях.

5. **Функції завантажника, що налаштовує.** Настраиваемый - До этапа выполнения программы в модулях описываются векторы переходов (внутри – и внешнемодульные), и константы, которые нужно переопределить. Загрузчик при просмотре этих записей

определяет абсолютные адреса гастраиваемых величин. Настройка адресных констант, Выделение памяти, Связывание, Собственно загрузка. Инфу для него готовит компилятор.

- выделение места для программ в памяти (распределение);
- фактическое размещение команд и данных в памяти (загрузка);
- разрешение символических ссылок между объектами (связывание);
- настройка всех величин в модуле, зависящих от физических адресов в
- соответствии с выделенной памятью (перемещение);
- передача управления на входную точку программы (инициализация).

6. **Дати визначення ініціалізації системи. Функції.** В виду того, что ядро ОС представляет собой совокупность взаимосвязанных модулей, имеющих связь друг с другом, процесс структурной организации модулей называется инициализацией. Инициализация системы - процесс создания ядра системы который включает не только перезапись программ, но и системных структур, обеспечивающих знания системы о ее параметрах. Программа - загрузчик. Small intro: ОС предназначена для удовлетворения нужд пользователей и должна содержать все необходимые пользователю модули или программы и поддерживать аппаратные средства. **Инициализация** – процесс структурной организации взаимосвязанных модулей,из которых состоит ядро ОС. Различают инициализацию *ядра* ,которая происходит по умолчанию и системы ,которая происходит по файлам инициализации.

7. **У чому відмінність меж максимальним паросполученням та досконалим.** Найти максимальное паросочетание - значит найти максимальное число ребер графа в которых не совпадают координаты вершин, либо найти максимальное число единиц матрицы у которых не совпадают координаты.

8. **Чому алгоритм «Корбато» краще FBn.** Алгоритм Корбата лучше алгоритма FBn, т.к. потенциально обладает большей производительностью т.к. распределяет задачи по очередям не по их приоритетам, а по признаку - абсолютная длина кода программы (чем меньше код тем выше приоритет). (В систему добавлен анализатор, который сразу размещает заявки в свою очередь, соответственно среднее время ожидания уменьшается.)

Що таке неоднорідна система? Як визначити степінь неоднорідності. Вычислительную систему можно рассматривать как совокупность неоднородных ресурсов с различной степенью детализации, а объекты планирования вычислений или заявки на ресурсы могут быть представлены как программы, процедуры, параллельные участки программ и т.д.

Понятие "неоднородность" системы можно рассматривать для трех случаев:

- В полунеоднородной системе, когда или задания, или ресурсы являются однородными или неоднородными. Число различных расписаний (до оптимизации) равно размещению из R элементов по N без повторения и без перестановки, то есть число возможных вариантов равно $\frac{R!}{(R-N)!}$ [21, 59, 134].
- В неоднородной системе и задания, и ресурсы являются неоднородными, и любое назначение задание—ресурс всегда является возможным (лишь имеют разные значения $\mathbf{T_e}$ и $\mathbf{T_c}$). Число различных расписаний (до оптимизации) равно размещению из R элементов по N без повторения, но с перестановками, тогда число возможных вариантов равно $\frac{R!N!}{(R-N)!}$ [72].

9. В строго неоднородной системе имеет место понятие совместимости ресурса—задания, при этом некоторые назначения ресурс—задание могут быть *невозможными*. Поэтому планирование для таких систем состоит из двух этапов: на первом этапе ищутся расписания возможных назначений максимальной мощности, а на втором этапе требуется найти оптимальное расписание из найденных на предыдущем этапе. Число возможных вариантов на первом этапе равно число различных и возможных расписаний (до оптимизации), или равно **числу** *максимальных паросочетаний* двух множеств **N** и **R** $V_1 \leq \frac{R!N!}{(R-N)!}$ [13, 40]**.** Число возможных вариантов на втором этапе $V_2 \leq V_1$.

10. **Функція мети задачі динамічного планування.** Под *динамическим* планированием понимается решение задачи распределения работ (составление расписания) и процессов в пространственных или пространственно—временных координатах во время выполнения вычислений и на том же оборудовании. Если планирование составляется на том же оборудовании на котором должно выполняться и во время решения задачи, то это планирование называется динамическим

11. **Сформулювати теорему про обов'язкові призначення.** С помощью анализа матрицы выделяют назначение, которое есть обязательным еще до нахождения базового решения. Теорема 1 :
Если в двудольном графе существует вершина, которой инцидентно только одно ребро, то вершины, инцидентные этому ребру, должны быть взяты в решение

12. **Що таке «свопінг»?** Свопинг - способ реализации многопрограммного режима работы на однопроцессорной машине

13. **Різниця між плануванням та розподіленням.** Задача — распределение (загрузка) (Task Allocation). Вычислительные процессы имеют *слабые* требования по предшествованию и отображаются несвязными графами. Это направление часто связывается с задачами балансирования и минимизации пересылок. При решении этих задач основой является определение — какой процесс (задание) будет выполняться на каком процессоре (ресурсе), а не определение порядка их выполнения [155]. При этом, в основном, решаются задачи пространственного распределения процессов (заданий).
Задача — планирование (Task Scheduling). Вычислительные процессы имеют сильные требования по предшествованию и отображаются связными ациклическими графами (DAG). При этом, к решению задач первого направления добавляется также определение порядка выполнения заданий. На этом уровне часто решаются задачи минимизации суммарного времени выполнения полного DAG на выделенных или имеющихся ресурсах.
Задачи по обоим направлениям, в общем случае, являются NP-полными или NP-сложными даже для двухпроцессорной системы [11] и имеют экспоненциальную временную сложность

14. **По який характеристиці можна судити про придатність розробленого алгоритму для роботи у динамічному режимі?** В большинстве случаев разработчики систем планирования реального времени используют статические алгоритмы и заранее определяют максимальный список заданий, допустив наихудший случай для получения статической управляющей таблицы (плана). Этот план фиксируется и используется для безусловного исполнения в динамическом режиме со следующими допущениями:

- все временные ограничения остаются неизменными на время выполнения плана;
- все задачи вкладываются в свое критическое время.

В других случаях при помощи приемов статического планирования создается статический список приоритетов для использования во время диспетчеризации самих работ в динамическом режиме.

15. **Етапи завантаження операційної системи.** Першим загружається BIOS. Він тестує обладнання, завантажує таблиці переривань. Завантажується завантажувач MBR, який визначає кількість розділів, чи є на них ОС та яка. Потім загрузка передається ОС. Визначаються апаратні засоби, обирається конфігурація. Після цього відбувається загрузка ядра, його ініціалізація. Потім ініціалізація системи.